Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Multi-objective Clustering of Gene Expression Data with Evolutionary Algorithms

A Query Gene Approach

Michael Calonder

Supervisor: Stefan Bleuler
Co-Supervisor: Dimo Brockhoff
Professor: Eckart Zitzler

Diploma Thesis DA-2006-01
Nov 11th, 2005 to Feb 20th, 2006

Abstract

Biologists are interested in the discovery of co-regulated genes since such genes are likely to share a common biological function. This work describes an evolutionary algorithm to find groups of genes in expression data that exhibit expression profiles similar to that of a given query gene. We are clustering over multiple data sets, such that the task is to maximize the *overall* co-expression. To this end, we treat the homogeneity of a certain gene group in every data file as an independent objective and employ a multi-objective evolutionary algorithm. In this context we also discuss three similarity measures, namely Euclidian distance, Pearson correlation and the ranked mean squared residue measure. We found that there is a clear trade-off between the abovementioned objectives. The query gene clustering algorithm is validated by showing that it is able to recover an already known cluster with 32 genes.
In a second step we introduce the cluster size as another objective that we require to be maximized. We found that this additional objective leads to an improvement in the quality of the clustering result and document this finding by 40 cases with different parameter settings.

Contents

Chapter 1

Introduction

1.1 Gene Expression

Many cells of most living organisms contain a set of codes called genes describing their regulation in form of one or more strands of the DNA molecule. The whole hereditary information encoded in the DNA molecule of an organism is called *genome*. The DNA resides in the nucleus of the cell, while most proteins are needed in the ambient cytoplasm, since it's where many of the cell's functions take place. Thus, DNA is copied into a more transient molecule called *RNA* and released into the cytoplasm. The concentrations of certain RNA molecules can be measured and are called *Gene Expression* levels.
A *gene* is a single segment of the coding region of the DNA that is transcribed into RNA. The RNA that codes for proteins is called *messenger RNA*, mRNA, and finally, after another intermediate stage (pre-mRNA), the information of the mRNA is translated into the target protein by the ribosomal complex, for more information see [15]. One method to measure the abovementioned expression levels are *microarrays*.

1.2 Microarrays and Data Matrix

Microarrays have been invented with the aim to measure the expression levels of a large number of genes simultaneously; they enabled biologists to study the regulation of genes and gene products of living organisms on a whole genome level. With the emerging microarray technology, it has become possible to measure the expression levels of a few tens of thousands of genes in one single experiment, which is in some cases the entire genome. Data resulting from such measurements is usually stored in a $m \times n$ gene expression data matrix E, where m is the number of genes considered in the experiment and n refers to the number of experimental conditions. Each element of E, e_{ij}, is a real value representing the abundance of mRNA for gene i under condition j, typically relative to a control experiment.
In the context of such data, one is interested in finding a subset of genes that are *co-expressed*; this means that the expression values of the genes of such a set are varying together over the columns of E. This is interesting from a biological point of view, since one often assumes that co-expressed genes are related to a common biological function [7].

1.3 Clustering and Biclustering

Clustering in general is the partitioning of a data set into subsets (clusters), such that the data in each subset is somehow similar – this similarity is often defined as some distance measure.
Clustering applied to gene expression data seeks to find groups of genes that are more or less co-expressed. This approach may be fair in cases where all the experiments belong together in some sense, but we cannot assume this for gene expression data while looking for co-expressed genes. We do not have a priori knowledge about which of the experiments are relevant to the biological function we are trying to identify. Additionally, expression data is made up of the more or less complete set of genes but very small portions of all possible conditions (experiments).
To overcome this problem inherent to clustering, one could require the clustering algorithm not only to select a subset of rows of E but also a subset of columns, see Figure 1.1.

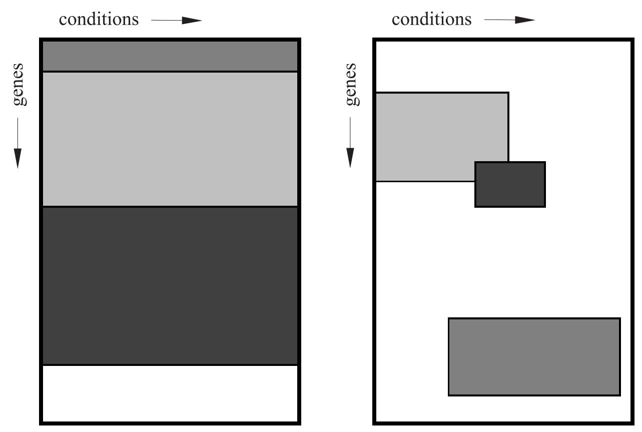


Figure 1.1: Clustering (left) vs. Biclustering. Traditional clustering searches a partition of all genes into k disjoint groups. Biclustering searches for one or a set of blocks containing a consistent local pattern. Three biclusters are shown. Note that is not generally possible to display several biclusters at the same time as contiguous blocks.

In a more general formulation, not even in the context of gene expression data, this clustering task is known as *biclustering* and goes back to the work of Hartigan [14]. Throughout this work when mentioning the term 'cluster', the object referred to may implicitly also be a bicluster. This will be clear from the context.

However, we are not only interested in finding a subset of genes, but also a subset of experiments, such that the selected genes need to be co-expressed on the selected experiments only. Obviously, the algorithm has now the possibility to drop experiments from the cluster that are likely to add unwanted noise. Any similarity measure between genes based on the available conditions becomes anyway context-dependent.
Additionally, biclustering allows rows and columns to be included in multiple clusters, and thus allows one gene or one condition to be assigned to more than one function. This added flexibility correctly reflects the biological reality [6].
Bleuler et al introduced a framework that is capable of biclustering, based on an

evolutionary algorithm which explores the search space on a global level [3].

There are cases where biologists have a specific gene under investigation and they are looking for other genes that are co-expressed with this one. We refer to such a gene as *query gene*; in some sense, a query gene defines a search pattern.
The algorithm developed by Owen et al, tailored to this specific problem (and also to case of multiple query genes), is called *gene recommender* algorithm [20].

1.4 Data sets and Multiple Objectives

We introduced the data matrix E in Section 1.2. This is one type of data that could be designated relevant regarding the clustering task but there are also other data types likely to contain relevant information (e.g., the Gene Ontology, GO). However, in this work we concentrate on the gene expression data type for simplicity and we usually work with two or more gene expression data sets.
Obviously, if there is more than one data set, things become more intricate. We want to look at this case in way that we now have more than one objective to satisfy. If there is a trade-off between the different objectives of the data sets, and we will see that this is the case, the problem is naturally one that demands multi-objective optimization. Therefore, as we introduce one homogeneity objective for each data set, we suddenly find ourselves in the field of *multi-objective optimization*. Our main motivation for using multiple objectives is the considerable advantage that we do not have to impose artificial information (in form of e.g. weights for parameterized aggregation) on the problem in advance. We are rather looking for a front that describes the trade-off between the different objectives, such that we may decide *afterwards* whether we are inclined to sacrifice a bit of one objective to receive a better performance in the other objective.

Additionally and apart from homogeneity maximization in the data sets we demand the cluster to be of a reasonable size ('reasonable' will be defined at some later point). To assure such an expedient size, we basically have two possibilities:

- we set a constraint to the minimum number of genes and chips we expect to be in the cluster, or
- we introduce a further objective favoring larger clusters.

1.5 Problem Formulation

In our approach we try to bring together the two already mentioned aspects, namely the

- Biclustering of expression data by an evolutionary algorithm (Bleuler et al),
- Gene recommender algorithm (Owen et al)

using a true multi-objective EA. We state our problem as follows: given a query gene, we search for a subset of genes and experiments that lead to the largest possible co-expression *for all of the given data sets*. Additionally, we may demand the cluster to be as large as possible.

1.6 Memetic Algorithms

The evolutionary algorithm we employ uses a local search strategy; the combination of an EA with a local search is called Memetic Algorithm [18]. They differ from other hybrid evolutionary approaches in that all individuals in the population are local optima, since after each variation step, a local search is applied. Both, MAs as EAs are population-based heuristic search methods and have been applied to a number of different areas, mostly combinatorial optimization problems. It is known that it is hard for a ”pure” EA to ”fine tune” the search in complex spaces [8]. Furthermore, it has been shown that a combination of global and local search is almost always beneficial [16].
MAs are inspired by Dawkin’s [8] notion of a meme. A meme is a ’cultural gene’ and in contrast to genes, memes are usually adapted by the individual who carries it before they are passed to the next generation. From the optimization point of view, it is argued that the success of an MA is due to the trade-off between the exploration abilities of the underlying EA and the exploitation power of the local search. Consequently, the balance between disruption and information preservation during variation is very important: on the one hand escaping local optima must be guaranteed, but on the other hand too much disruption may cause a loss of important information gained in the previous generation [23].

Chapter 2

Related Work

This chapter gives a brief overview on related work that has influenced this research. In the following we will sketchily focus on three aspects: clustering, multi-objective optimization and data types.

2.1 Clustering, Biclustering

There are many approaches to clustering and partitioning techniques in gene expression analysis. Many of them are summarized in a review by Hand and Heard [11], to name but a few, there are hierarchical clustering approaches, nonmodel-based partitioning methods as self-organizing maps or techniques based on a singualar value decomposition of the expression matrix. However, in this section we want to focus on just a small part of this body of work, namely clustering and biclustering. For details regarding clustering and biclustering, see Section 1.3.

An approach to biclustering of expression data has been presented by Cheng and Church [6]. They introduced both, a straightforward and a more sophisticated algorithm for biclustering.
The former simply starts from the full expression matrix and subsequently deletes those nodes[1] which decrease some mean squared residue g, a measure for the inhomogeneity of the bicluster, the most. When g is below some given constraint value δ, the algorithms stops. δ was found to be a powerful parameter to fine-tune the similarity requirements.
The sophisticated and more efficient version, starts also from the full expression matrix and proceeds in three steps: multiple node deletion, single node deletion and node addition. The main difference to the straightforward version lies in the criterion raised to remove and add nodes. They proofed a theorem that allows to remove and add nodes without the need of recalculating g for every node. Furthermore, there are cases where it is beneficial to remove multiple nodes from the bicluster at once until the matrix reduces to a size that single node deletion can handle efficiently. This is realized in the multiple node deletion procedure.
However, this method is designed for biclustering but in this form, it is not

[1]We use the term *node* if we don't want to specify whether we are concerned with a gene or an experiment/condition.

applicable to a query gene problem.

Bleuler et al investigated how this procedure, among others, can be integrated in a global optimizer, namely a framework for evolutionary algorithms (EAs), cf. [3]. They coupled an EA with a local search procedure implementing the efficient strategy from Cheng and Church. If possible, the local search improves in every generation the actual individual that the enclosing EA suggests and seeks to evaluate. They demonstrated that the quality of the outcome can be substantially improved by this hybrid approach. While the evolutionary algorithm alone has failed to produce results comparable to those generated by Cheng and Church's algorithm, both algorithms together clearly outperformed the pure greedy strategy in terms of the quality of the resulting biclustering.
We will be using the above problem setting at a later point for both to check and to extend our newly developed algorithm.

A distinct approach toward the discovery of similarities in expression data has been published by Owen et al; they present in [20] the so-called *Gene Recommender Algorithm*. It aims at finding genes that have a function similar to that of some *given genes*. Usually, the members of the given set of genes are a priori known to have a closely related function themselves. These genes are referred to as *query genes*.
The work of Owen et al belongs to a body of research called feature selection. It is the only source known to the authors that pursues this type of clustering, where one is interested in finding genes given some pattern. There is some information on related, but, in view of our task, different investigations in [20].
Basically, we are concerned with a similar but slightly more abstract question than the Gene Recommender Algorithm because we are not only interested in up-regulated genes but in co-regulated genes in general. In contrast, we take multiple datasets into account and use the EA framework from Bleuler et al [3] briefly mentioned above to implement a new local search strategy (called T2QGS, type 2 query gene search, the name will become clear later). Our approach can thus be viewed as the combination of a globally acting optimizer and a local search, similar to [3] from above.
Another, minor difference lies in the number of query genes: we use exactly one where the Gene Recommender Algorithm treats more than one query gene. The extension of the framework to many query genes is not expected to bring about major problems, especially not for we kept this topic in mind during the implementation of the T2QGS algorithm.

At this point, it is natural to inquire after finer differences between the approach from Owen et al and ours. In order to achieve a clear distinction, we briefly outline the idea of the Gene Recommender Algorithm. It basically follows the five steps given below. Let Q be the set of query genes.

1. Normalize the expression data in E. This is done by first assigning each element the ranks within its row and then normalizing these ranks to the interval $[-1, 1]$. E' is the matrix of the normalized values.

2. Assign a Z-score to every condition. This score prefers conditions that show

a tight clustering over Q in the sense of variance and extreme mean values of E' of Q.

3. Collect the conditions with the highest Z-scores (which are deemed relevant to the query genes Q) in the set ϵ. The threshold defining the number of topmost conditions that should be included in ϵ is set by some appropriate procedure.

4. Assign a score $S_G(m)$ to every gene $m \notin Q$. This score measures the extent to which gene m matches the query Q and is calculated based on the conditions in ϵ only. This yields a hit list, having the gene with highest $S_G(m)$ at the top.

5. Truncate hit list such that it contains the same number of genes as G does.

2.2 Multiple Objectives

In the introduction we explained our interest in multi-objective optimization. In the context of clustering of data, not specifically expression data, Handl and Knowles investigated the effect of multiple, complementary objectives and found that there is promising potential to such an approach [12]. However, their work differs from ours mainly in two aspects: Handl and Knowles suggest a method

- for *clustering* (instead of *biclustering*).
- to cluster *one* data matrix (instead of several).

They applied an EA to optimize two conceptually orthogonal objectives simultaneously, and for comparison, each objective has been optimized independently. They found *objectively* better solutions for the multi-objective optimization scenario and thus they concluded that it is advantageous to optimize several objectives concurrently. – A result that strengthens our intent to tackle the query gene search problem by multi-objective optimization.

2.3 Data Types

Using as much information as possible for the clustering task is desirable and also indirectly related to multi-objective optimization. Data types like metabolic pathways, PPI (protein-protein interaction), sequence analysis, Gene Ontology[2], and others are such candidates. These could be e.g. included as further objectives.
One step into this direction has been made by Speer et al [23]. They introduced a memetic co-clustering algorithm called MST-MA (maximum spanning tree memetic algorithm) that incorporates gene expression data and the GO. MST-MA calculates on each of these two data types a distance, given a pair of genes. In the case of the GO this is some semantic distance measure where for the expression profiles this is the Pearson correlation between the two expression profiles. The MST-MA has been designed for single objective optimization and accordingly the

[2]In general, ontologies offer a mechanism to capture knowledge in a sharable form that is also processable by computers. They provide a set of vocabulary terms that label domain concepts and at the same time terms are placed within a structure of relationship.

two distances are merged into one distance that serves as an objective value. This has been achieved by parameterized aggregation (or rather a weighted sum).
Apart that MST-MA and our approach differ in the GO data type, there are at least two other differences:

- **Representation.** MST-MA represents a solution in form of a tree and is therefore left with a tree partitioning problem. The idea is to find a set of tree edges to delete, such that the resulting unconnected components determine the clustering. As documented later, we rely on a bit string representation.
- **Number of objectives.** MST-MA embodies single objective optimization achieved by parameterized aggregation of the distances where we pursue true multi-objective optimization (see also Section 2.2).

At a future date we intend to work with several different data types, but at the moment we only operate on one type of data, namely expression data. From a biological point of view, this is interesting as well, because already the expression data sets show a trade-off among each other, as we will see later.

Turning now to expression data, a further research based on *more than one data set* has been conducted by Bleuler and Zitzler [5]: they pursued an order preserving clustering approach over multiple time course experiments. – Usually, the measurements of different experiments are mixed together in a single gene expression matrix. Hence the information about the shared identity of the experiments of the data sets is eliminated. To avoid this loss of information, they operate with the data sets separately. In order to exploit the information therein, they adopt the concept of the *order preserving submatrix*, OPSM [1] and cluster on transformed data. Furthermore, a new score that measures the degree of order preservation is introduced. The measure is basically a combination of the mean squared residue score [6] and the OPSM [1] concept. Again, the node deletion and addition strategy from Cheng and Church is implemented as a local search in an EA framework, similar to [3]. The local search first removes genes until the homogeneity constraints are met *for all data types (files)* and in a second step all genes that can be added without increasing the homogeneity of the cluster are added to the cluster.
Our approach is quite similar to this one. The T2QGS algorithm is embedded in the same framework that is already able to handle a number of data files separately and we basically introduced a new local search. We also make use of the OPSM concept which we detail below, but we additionally explore two other distance measures, the straight forward Euclidian distance as well as the Pearson correlation.
The framework already provides the option for both, multiple objectives and biclustering. Traditional clustering can be achieved as a special case of biclustering, where the constraints for the number of experiments to include is set to the number of experiments for each file. When dealing with multiple objectives, the Selector for PISA (see below) has also to be chosen appropriately. For our purposes, we used IBEA, an indicator based evolutionary algorithm [25].

Chapter 3

Problem Outline

In a first part we are going to define two groups of problems relevant to this work, SOSM/SOHM and T1QGS/T2QGS respectively.
The section on the QGS problems, but also Section 1.3 call the attention to the need for a distance measure between two genes. Therefore we expand in a further section on three such measures, namely the Euclidian distance, the Pearson correlation and the ranked mean squared residue (MSR) measure.
For the remainder of this work, let G denote the subset of genes of E and C the subset of experiments of E defining the bicluster $\{G, C\}$.

3.1 The SOSM and SOHM Problems

After having introduced the notion of distance or similarity between genes, we are ready to mathematically describe the problems.
The SOSM and SOHM problems are related to each other in an opposed sense. SOSM asks to maximize the size of a bicluster, constrained to some maximum dissimilarity where the task in SOHM is to maximize the similarity constrained to some minimum cluster size. The former problem has already been implemented in the EA framework [3, 5] such that it will basically serve for validation in principle. Extending the framework to enable it to handle the latter case is the minor of two parts of this work.

3.1.1 SOSM Problem

SOSM stands for $\underline{\text{s}}$ingle $\underline{\text{o}}$bjective $\underline{\text{s}}$ize $\underline{\text{m}}$aximization. We already addressed this single objective problem in Section 2.1 in an informal way and we now want to define it quantitavely. The *mean squared residue* (MSR) of some bicluster $\{G, C\}$ indicated by the subsets G and C of genes and conditions respectively, is given by

$$g(G,C) = \frac{1}{|G|\,|C|} \sum_{\substack{i \in G \\ j \in C}} (e_{ij} - e_{iC} - e_{Gj} + e_{GC})^2 \tag{3.1}$$

where

$$e_{Gj} = \frac{1}{|G|} \sum_{i \in G} e_{ij}, \quad e_{iC} = \frac{1}{|C|} \sum_{j \in C} e_{ij} \tag{3.2}$$

are the mean column and row expression values for $\{G, C\}$ and

$$e_{GC} = \frac{1}{|G|\,|C|} \sum_{\substack{i \in G \\ j \in C}} e_{ij} \tag{3.3}$$

is the mean column and row expression value over all cells contained in the bicluster. Using this notation we can now quantify the SOSM problem. Following [6], the optimization goal is to find a bicluster of maximum size that does not exceed some MSR value δ.

$$\begin{array}{lrcl} \text{max} & f & = & |G| \cdot |C| \\ \text{subject to} & g(G,C) & \leq & \delta \\ & \{G,C\} & \in & \mathcal{X} = 2^{\{1,\ldots,|G|\}} \times 2^{\{1,\ldots,n\}}, \end{array} \tag{3.4}$$

where $\mathcal{X}$ denotes the search space consisting of all possible combinations of genes and experiments. Note that the size of the search space $|\mathcal{X}| = 2^{|G|+|C|}$ is exponential in both, number of genes and number of experiments. Typical values for $|G|$ are between 1'000 to 25'000 and $|C|$ usually ranges from 10 to about 100. Obviously, this induces a huge search space. Additionally, the problem has been shown to be NP-complete [6].

3.1.2 SOHM Problem

SOHM stands for <u>s</u>ingle <u>o</u>bjective <u>h</u>omogeneity <u>m</u>aximization. In terms defined in the previous section, the problem reads

$$\begin{array}{lrcl} \text{min} & g(G,C) & & \\ \text{subject to} & |G| \cdot |C| & \geq & \sigma_{min} \end{array} \tag{3.5}$$

where σ_{min} denotes a lower bound on the cluster size.

3.2 The QGS Problems

QGS stands for <u>q</u>uery <u>g</u>ene <u>s</u>earch. There are basically two such problems, T1QGS (type-1 QGS) and T2QGS (type-2 QGS) which fundamentally differ from the two other problems above (SOSM and SOHM) in that they are clustering around a *given* query gene. The two types of QGS problems arise naturally from the fact that one is given some query gene. See Table 3.1 for the definitions. Even though this work is mostly concerned with the T2QGS problem, we will also compare the results from T2QGS runs against results from the SOMS runs.

In the T2QGS definition we stated that the cluster size σ may be included as a further objective or not. There is still a third possibility: we could directly integrate the size into the calculation of the objective values $f_1, \ldots, f_n$. For instance, we could reassign the objective values

$$\forall 1 \leq i \leq n \; : \; f_i \longleftarrow \frac{f_i}{\sigma} \, .$$

Approaches to include the size in the objective assignment of this type are left aside in this work since we are mainly interested in true multi-objective

Name	*Definition*
T1QGS	This is the identical problem as SOSM plus the requirement that the query gene must be part of the bicluster. Consequently, the query gene may lie *anywhere* inside the cluster with respect to some distance measure. We call this a *type 1 problem.*
T2QGS	For a given number of data sets n, select a subset of genes G and n subsets of experiments C_i, $1 \leq i \leq n$, such that the *overall* similarity is maximized[a]. The term 'similarity' is quantified as total distance from a query gene to the remaining genes in the cluster. Additionally, we may include a further objective that favors large clusters. Later, it will be stated for every case whether this objective has been included or not. The analytical form is given below. $$\begin{array}{lrcll} \min & f_1 & = & \mathrm{dist}\,(G, C_1) & \\ \vdots & \vdots & \vdots & \quad\vdots & \\ \min & f_n & = & \mathrm{dist}\,(G, C_n) & \\ \min & f_{n+1} & = & \sigma^{-1} & \\ & & & & \\ \text{subject to} & \lvert G \rvert & \geq & G^{min} & \\ & \lvert C_i \rvert & \geq & C_i^{min} & \forall\, 1 \leq i \leq n \end{array}$$ where $\sigma := \lvert G \rvert \cdot \lvert C \rvert = \lvert G \rvert \cdot \sum_{i=1}^{n} \lvert C_i \rvert$ and $\mathrm{dist}\,(G, C_i)$ is the total distance between all genes in G and the query gene, measured on experiments C_i. [a]Note that the genes are the same for all data sets where the selected chips may differ for each dataset.

Table 3.1: Definition of the QGS problems. See Figure 3.1 for an illustration.

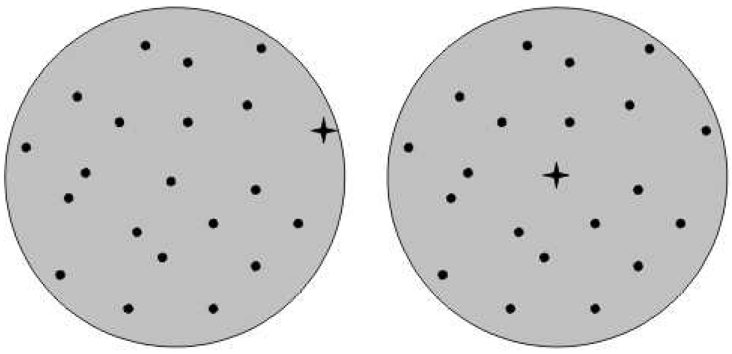


Figure 3.1: (Left) Type 1 problem, the query gene (✦) may lie anywhere in the cluster. (Right) Type 2 problem, the query gene (✦) is in any case somewhere in the center of the cluster.

optimization. Furthermore, note that T2QGS implicitly includes the size σ as a constraint, since both $|G|$ and $|C_i|$ are constrained and $\sigma = \sigma(|G|, |C_i|)$.

We are now going to discuss three distance measures in detail. Because we don't want to tear apart the discussion on these, we first introduce the OPSM concept that underlies the ranked MSR measure.

3.3 The OPSM Concept

The term OPSM already appeared in section 2.3; we will now define it more precisely. A submatrix is called *order-preserving*, if there exists a permutation of its columns under which the sequence of values *in every row* is strictly increasing. In the case of expression data, an OPSM is determined by a set of genes G and a set of experiments C, such that within C the expression levels of all the genes in G have the same linear ordering [1]. Figure 3.2 illustrates this idea.

0.34	0.01	0.23	0.21
0.43	0.69	0.49	0.58
0.10	0.12	0.19	0.30
0.13	0.22	0.34	0.52
0.71	0.19	0.77	0.34

$\xrightarrow{\text{ranking}}$

4	1	3	2
1	4	2	3
1	2	3	4
1	2	3	4
3	1	4	2

0.34	0.01	0.23	0.21
0.43	0.69	0.49	0.58
0.10	0.12	0.19	0.30
0.13	0.22	0.34	0.52
0.71	0.19	0.77	0.34

$\xrightarrow{\text{ranking}}$

2	1	2	1
1	2	1	2
1	2	1	2
1	2	1	2
2	1	2	1

Figure 3.2: (top row) A gene expression matrix with the expression values and the related ranks within the row. (bottom row) The same situation as above, but now the two data sets (each with two columns) are treated separately. Effect: In the second case exists one extra gene that can be attributed to the OPSM, compared to the first case.

3.4 Distance Measures

Since a distance measure represents the actual link between a bicluster and its resulting fitness values, we especially strive for a deeper understanding of such quantizations. We have investigated three different measures for distance or rather for similarity, since similarity is what we are finally interested in; these are

- Euclidean Distance
- Pearson Correlation
- Ranked MSR (Mean Squared Residual)

From the considerations in the following three subsections, we expect the best performance from the ranked MSR measure. A description of the data sets is given in Section 5.1.

3.4.1 Euclidean Distance

The Euclidian distance between two genes i and j corresponds to the Euclidean distance between their row vectors of the expression matrix. If gene j is set to the query gene q, this reads

$$d_e(i,q) := \left(\sum_{k \in C} (e_{ik} - e_{qk})^2 \right)^{\frac{1}{2}} .$$

For a randomly selected query gene we plotted the Euclidean distances over the sorted gene indices for three different data sets, see Figure 3.3.
On real data (Figure 3.3a and 3.3b) the distances are distributed differently for every data set[1] as expected. Since the two data sets in RAND_N are sampled from the same distribution, we would anticipate that the distance plots for these two sets coincide, especially for a large number of samples as we have in our case. However, the upper part of Figure 3.3c shows that there is a discrepancy between the two curves. Since this artifact has been observed for other, also random data sets, we interpret this fact as a tendency of the Euclidean distance measure to introduce some kind of bias.

3.4.2 Pearson Correlation

As an attempt to overcome the aforementioned uncertainty about the distance measure, we conducted the same experiments as before except that we replaced the Euclidean distance measure with the Pearson correlation. This can be justified insofar that we are not interested in absolute values (at least not in context with expression data) but rather their relative ranks. Correlation is a measure that does not account for differences in the mean of the data.

[1] Please note the subtle distinction between *data set* in general, referring to some experiments that form a set and *input data set* including one or more (in our case usually two) data sets that form the input information to the algorithm.

As before in terms of a distance from a gene i to the query gene q, the correlation based distance is

$$d_c(i,q) = -\frac{1}{n-1}\,\frac{1}{\sigma_i\ \sigma_q}\ \sum_{j\in C}(e_{ij}-e_{iC})(e_{qj}-e_{qQ}),$$

where σ_i and σ_q denote the unbiased sample variances of the corresponding gene expression profile and e_{iC} as defined in Eq. (3.2). The minus sign accounts for a consistent perception of distance, since the other two distance measures associate small values with a high similarity.

Figure 3.4c shows the previously expected result for random data. There is another qualitative issue to notice. If one could recognize a more or less sharp ”knee” (best in both data sets for the same genes, which cannot be observed from this kind of plot), we would know where to cut off the sorted list of genes and would have found our cluster. We cannot observe such a knee from Figure 3.3 whereas in Figure 3.4a or Figure 3.4b one is tempted to recognize such an artifact. However, Figure 3.4c refutes that this artifact is meaningful, since we can find the same behavior in randomly sampled input data.
There is no proof that our algorithm needs such a behavior of the distance measure. But in general, one should avoid to interfere with the algorithm by introducing a bias through an inept choice of distance measures.

3.4.3 Ranked MSR

We have not yet considered the robustness of the distance measures regarding noise. Expression data arise from biological experiments that are likely to introduce measurement uncertainties. A convenient way to overcome this problem is to rank the data within each row. Let $r_{i,j} \in \{0, |C|-1\}$ be the rank of the i-th row of E and scale the ranks according to

$$r'_{i,j} = \frac{r_{i,j}}{|C|-1} \quad \in [0,1].$$

Then, the ranked MSR distance d_r is defined by

$$d_r(i,q) = \frac{1}{|C|}\sum_{j\in C}\left(r'_{i,j} - r'_{q,j}\right)^2. \tag{3.6}$$

We find that the Euclidian distance *on this ’stabilized’ data* passes the sanity check of the coinciding curves on random data (Figure 3.5c). Since this measure does not account for differences in the mean value as the neither the Pearson correlation does, we expect them to serve the purpose equally well (except in the presence of noise). In 3.4.4 we will find that they show some degree of correlation, as expected.

This conception of similarity goes nicely together with the OPSM concept introduced in Section 3.3. Altogether, we employ a scoring function that proceeds in the following steps [5]. Suppose we want to score the submatrix $D \subseteq E$ implied by the bicluster $\{G, C_i\}$:

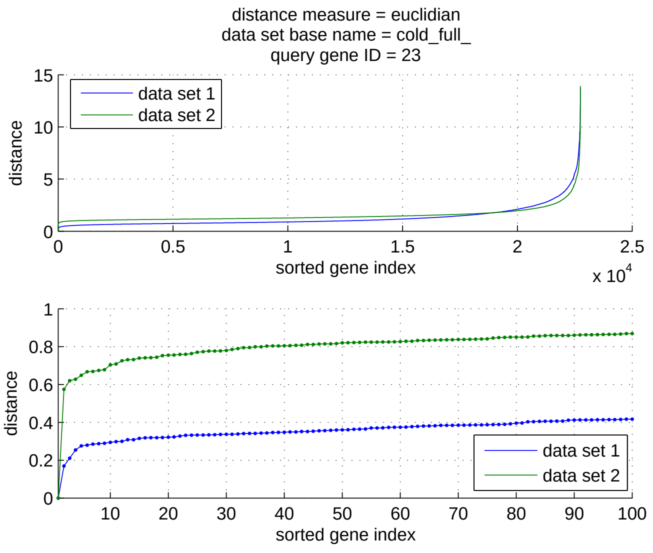
1. Rank the values in D per row and replace the expression values with their (scaled) ranks.

2. Apply the mean squared residue score to the transformed values to receive the score for the submatrix D.

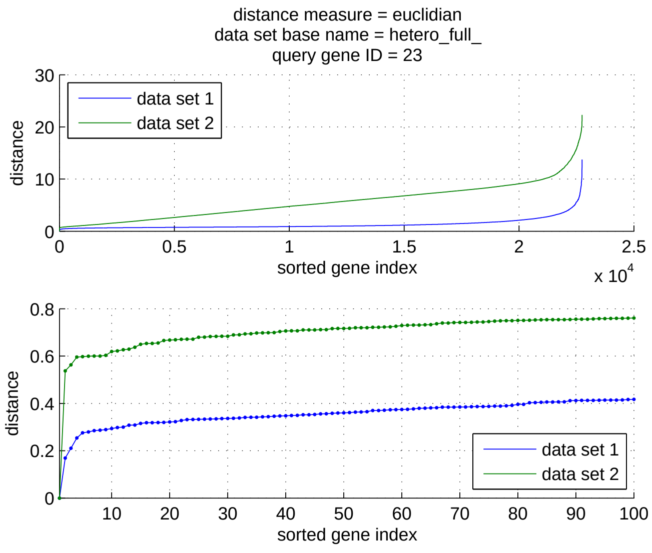
The scoring scheme has a remarkable property: a score of zero is equivalent to D being an OPSM ('perfect' clusters). However, in most of the relevant simulations we made use of this measure.

3.4.4 Comparing Distance Measures

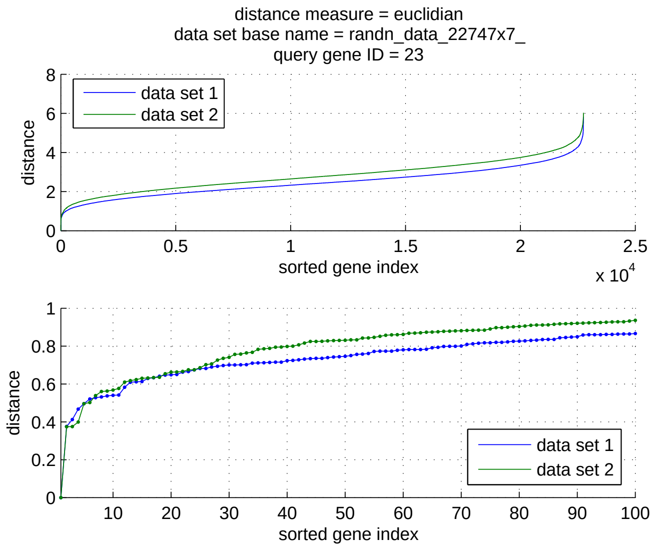
Here we want to show that the distance measures are not correlated among each other with the exception of the Pearson correlation and the ranked MSR distance measure. The three distance measures are compared in Figure 3.6. If all points would lie on the "$y = x$"-line, this indicated a perfect agreement of the measures and therefore an indifference regarding our algorithm, since we are acting on an ordinal, not quantitative scale. To create such a plot that compares measure 1 (vector of distances d_1) against measure 2 (d_2), sort d_1 and d_2 in ascending order resulting in the permutations of indices p_1 and p_2, respectively. Starting with the first element of p_1, $p_1(0)$, find the *value* $p_1(0)$ in p_2 and store the *index* in i. Draw a point at $(p_1(0), i)$ and continue with the next element of p_1.



3.3a COLD_FULL input data set.

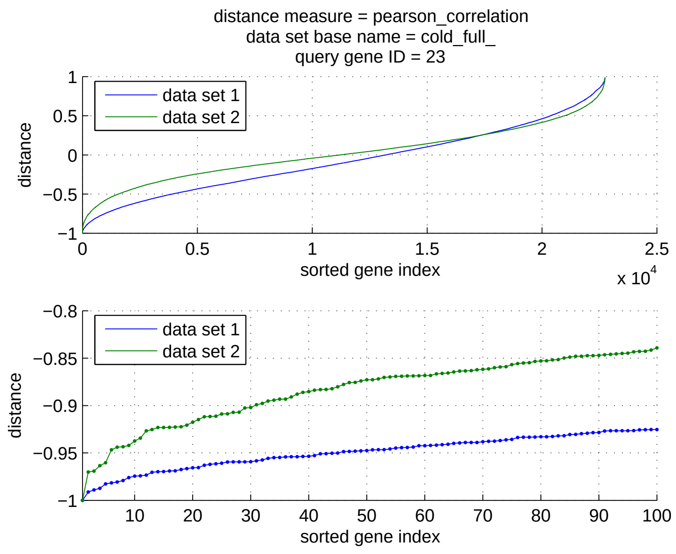


3.3b HETERO_FULL input data set.

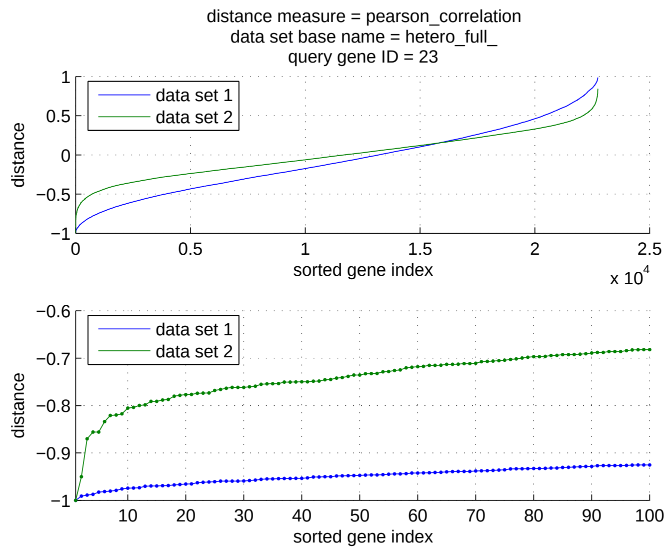


3.3c RAND_N input data set.

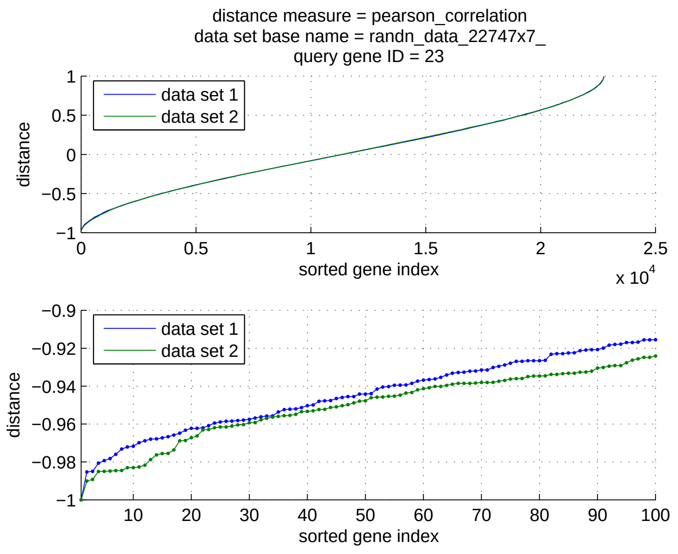
Figure 3.3: Euclidean distances for three different input data sets. The subfigures show in their upper part the overall behavior of the measure where the lower part shows only the first 100 genes. The domain up to 100 genes is of peculiar importance since it is known from biology that it is very unlikely to encounter biologically relevant clusters with hundreds of genes (let alone with thousands of genes).



3.4a COLD_FULL input data set.

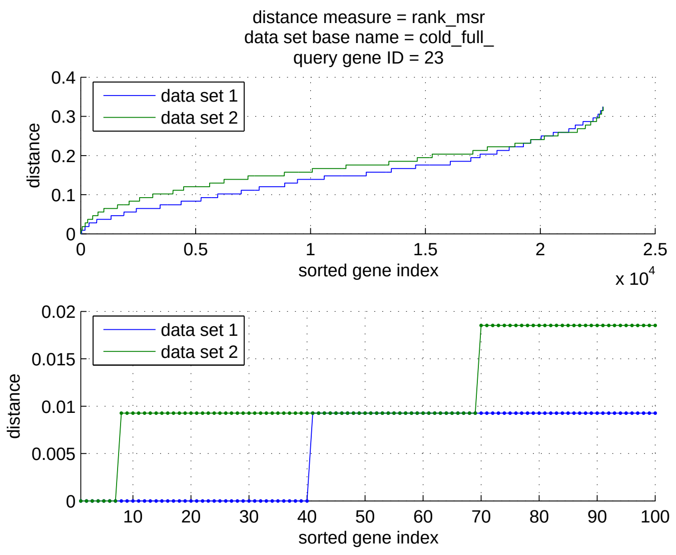


3.4b HETERO_FULL input data set.

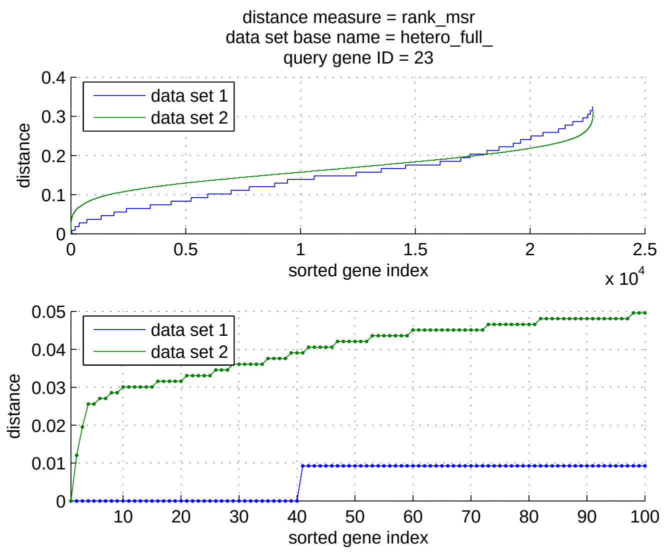


3.4c RAND_N input data set.

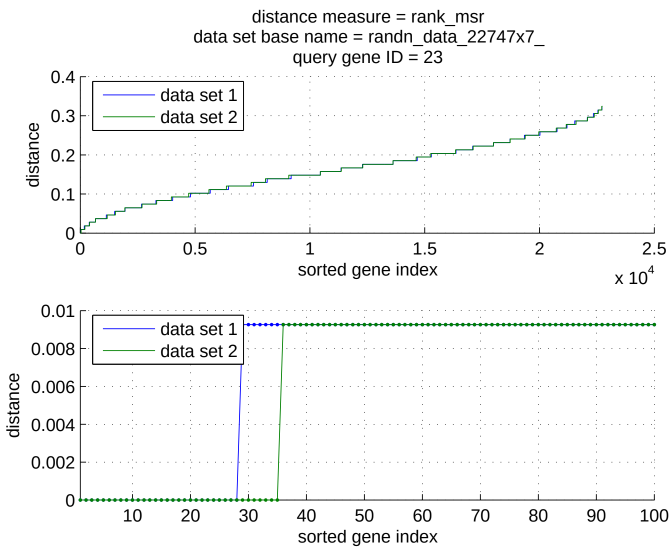
Figure 3.4: Negative Pearson correlation for three different input data sets.



3.5a COLD_FULL input data set.

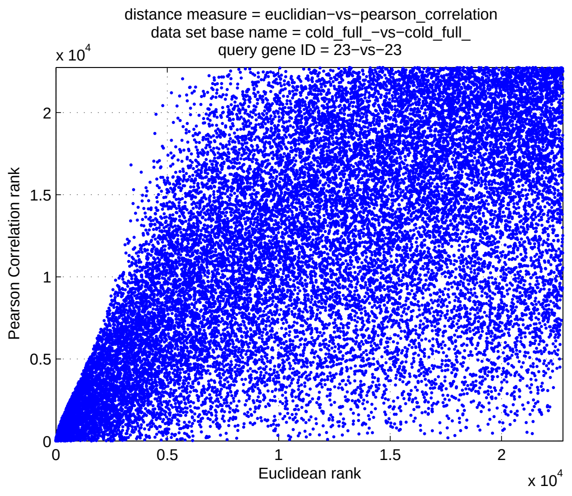


3.5b HETERO_FULL input data set.

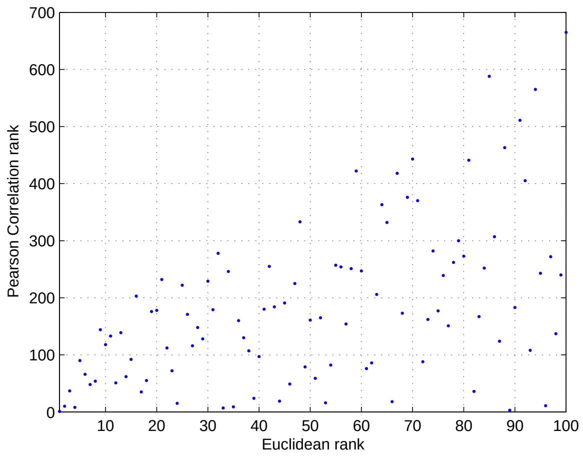


3.5c RAND_N input data set.

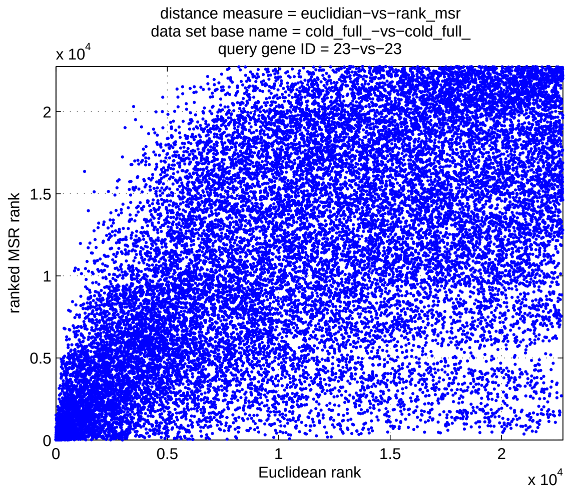
Figure 3.5: Rank MSR distance for three different different input data sets. One conspicuous feature of the plots is that there are some discrete distance levels; their number does not exceed 37, even though there are $6! = 720$ possible levels.



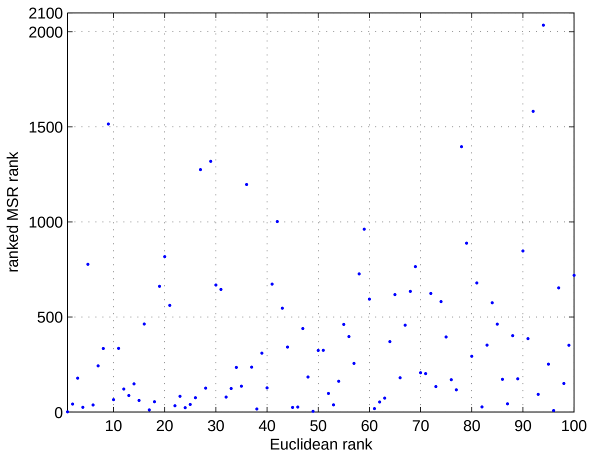
3.6a Euclidean vs. Correlation



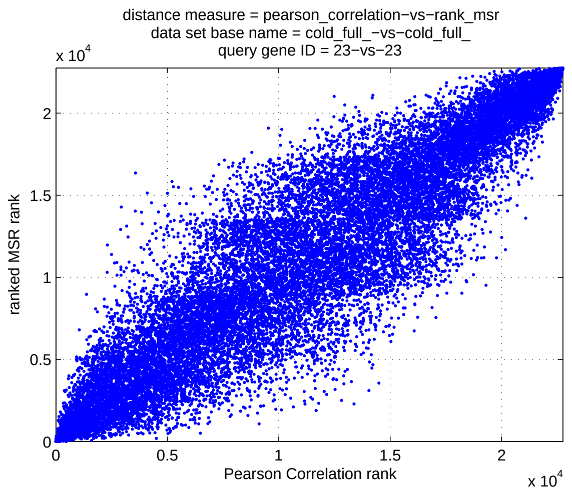
3.6b Euclidean vs. Correlation (detail)



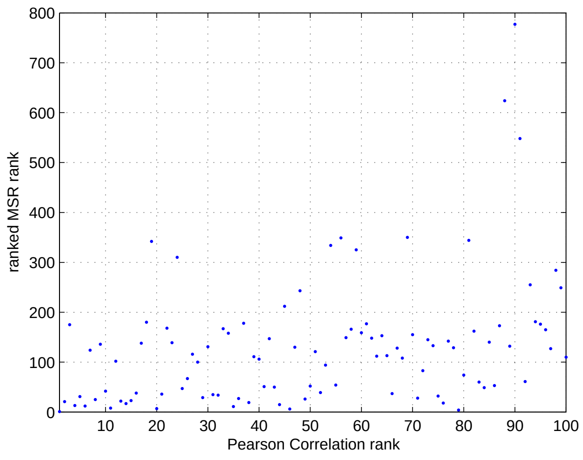
3.6c Euclidean vs. ranked MSR



3.6d Euclidean vs. ranked MSR (detail)



3.6e Correlation vs. ranked MSR



3.6f Correlation vs. ranked MSR (detail)

Figure 3.6: Exemplary comparison of three distance measures on the data set COLD_FULL. The results are principally identical for the other data sets.

Chapter 4

Algorithms and Implementations

The first two sections abstractly introduce the evolutionary algorithm and the PISA framework. A third extended chapter is dedicated to the local search procedure since it is, together with some extensions of the framework, at the heart of this work. We will give a few algorithms in detail but will forgo any source code in favor of implementation concepts.
We will then introduce a mutation operator that flips the same number of ones to zeros as vice versa and compare it later against the standard, independent bit flip operator. Furthermore, we describe a method to constrain the search radius of the local search, because it tends to gather individuals in identical local optima (especially on multi-modal search landscapes).

4.1 The Evolutionary Algorithm

We briefly want summarize points concerning the evolutionary algorithm. The following list gives a few cornerstones of the EA encapsulating the T2QGS local search strategy. A schematic view of the EA is depicted in Figure 4.1.

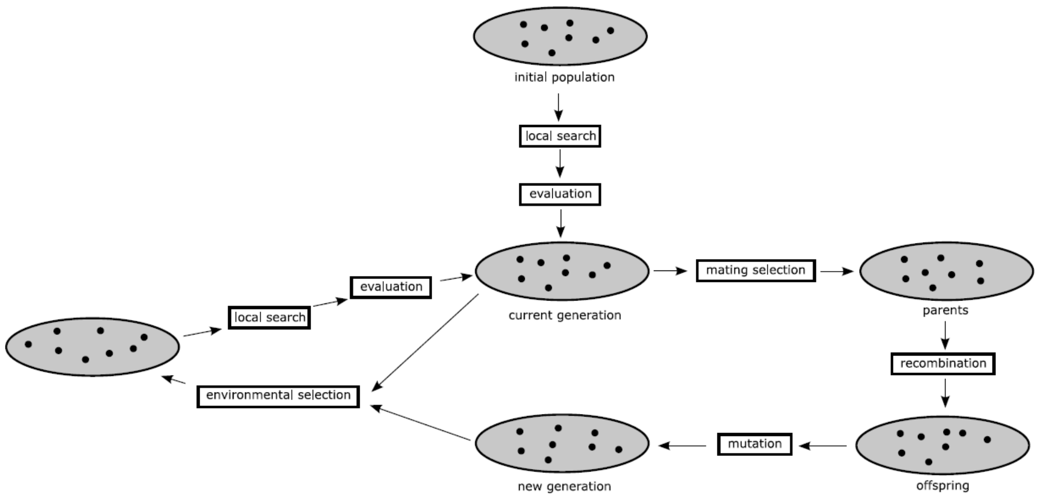


Figure 4.1: Schematic view of an iteration in an evolutionary algorithm.
Adopted from [4].

- **Solution representation.** An individual is encoded as two bit strings of the lengths $|G|$ and $|C|$ respectively, which is straightforward. There are other suggestions towards the representation of a solution, like minimum spanning trees [23] or group oriented encodings [9].
- **Fitness Assignment.** The local search is delegated to assign the fitness values to an individual. Since the new local search `T2QGS` will be extensively discussed at some later point, we will not go into details here.
- **Mating Selection.** The mating selection scheme is implemented as a tournament selection and is usually of size 2.
- **Variation Operators: Mutation and Recombination.** There are two mutation strategies we employed. One was independent bit mutation, where and arbitrary bit is flipped with some fixed probability. However, we considered a method that flips the same number of bits from 1 to 0 as from 0 to 1 for it appeared a more adequate operator for variation. For the recombination part, we use an uniform crossover strategy that assigns a bit from the first individual to the second with a random but fixed probability and vice versa.
- **Environmental Selection Scheme.** Selection is entirely left up to IBEA, for details see [25].

In the next section will be briefly introducing the PISA framework that constitutes the EA.

4.2 The PISA Framework

The framework for the EA called PISA is a platform and programming language independent interface for search algorithms [2] completely implemented in C++. It consists of two independent modules, the Variator and the Selector that communicate via text files. This separation in a problem specific and an optimization algorithm specific part is especially beneficial in multi-objective search algorithms, since it allows the application engineer to test several optimization algorithms in a plug-in fashion, see Figure 4.2. In this work, we are concerned with the problem specific part where the `T2QGS` algorithm is implemented. For the optimization algorithm we employed IBEA for all runs.

4.3 Local Search

In the introduction we presented the fundamental notion of a memetic algorithm: it tries to combine the exploitation strength of a local search with the exploration power of a global optimizer, the EA. This entails that the EA is concerned with locally optimal solutions only but potentially, the local search is too strong and its neighborhood needs to be restricted.

From a global point of view, this works as follows (Figure 4.3). To indicate to the system that we wish to solve a problem with, say, the `T2QGS` local search, we introduce a parameter in `ebica_param.txt` named `evaluation_type` that

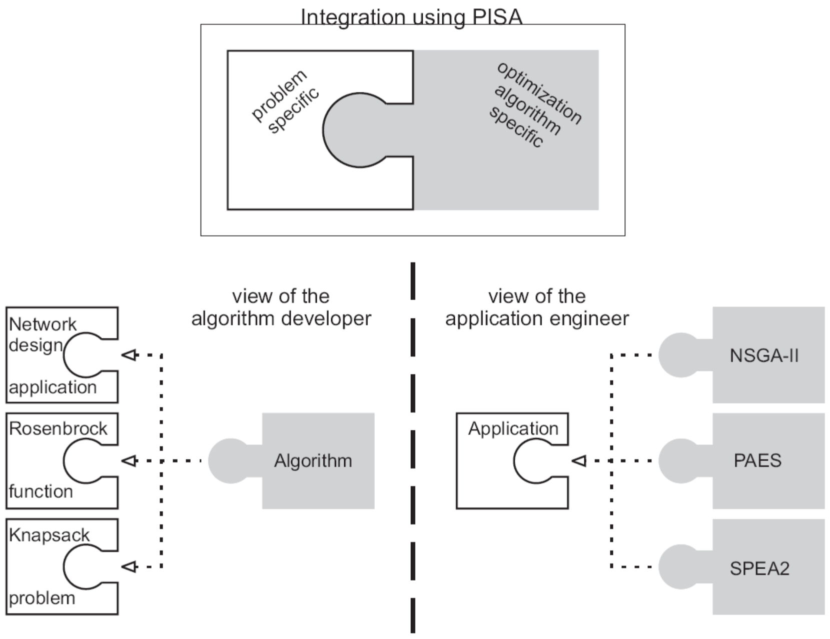

Integration using PISA
problem specific
optimization algorithm specific
view of the algorithm developer
view of the application engineer
Network design application
Rosenbrock function
Knapsack problem
Algorithm
Application
NSGA-II
PAES
SPEA2

Figure 4.2: Schematic view of the PISA framework.

can be set to the name of the method, in this example `qgs_t2_ls`. Based on this parameter, the framework allocates a local search object of the desired type. For the individual, it suffices to hold a static reference to a `local_search` object which receives its dynamic type through the aforementioned initialization. The invocation of the correct functions can then be abandoned to polymorphism.

The PISA framework in its modular architecture perfectly allows for the implementation of a new local search strategy in a plug-in fashion. PISA is innately able to deal with multiple objectives and since the local search is accountable for the assignment of fitness values, we only need to focus on the implementation of the local search that must be able to assign multiple objectives (apart from all other functionality of course). The following two sections describe the SOHM and `T2QGS` strategies.

4.3.1 SOHM Local Search Strategy

Section 3.1.2 outlined the optimization goal of the SOHM problem. We tackled this problem by implementing a new local search strategy as an alternative to the local search that solves the SOSM problem. We will now consider some details.

Concerning the implementation, we could build upon previous work: given the implementation of the SOSM in PISA [2, 3, 5], we mostly needed to re-implement parts of the local search (`moop_rank_ls::ls_so_size`). This local search resembles the three-step strategy of SOSM insofar as that it also follows the idea of Cheng and Church, mentioned in Section 2.1. Furthermore, it also guarantees that only feasible solutions result from the local search.
We are finally left with the implementation of three `moop_rank_ls` member func-

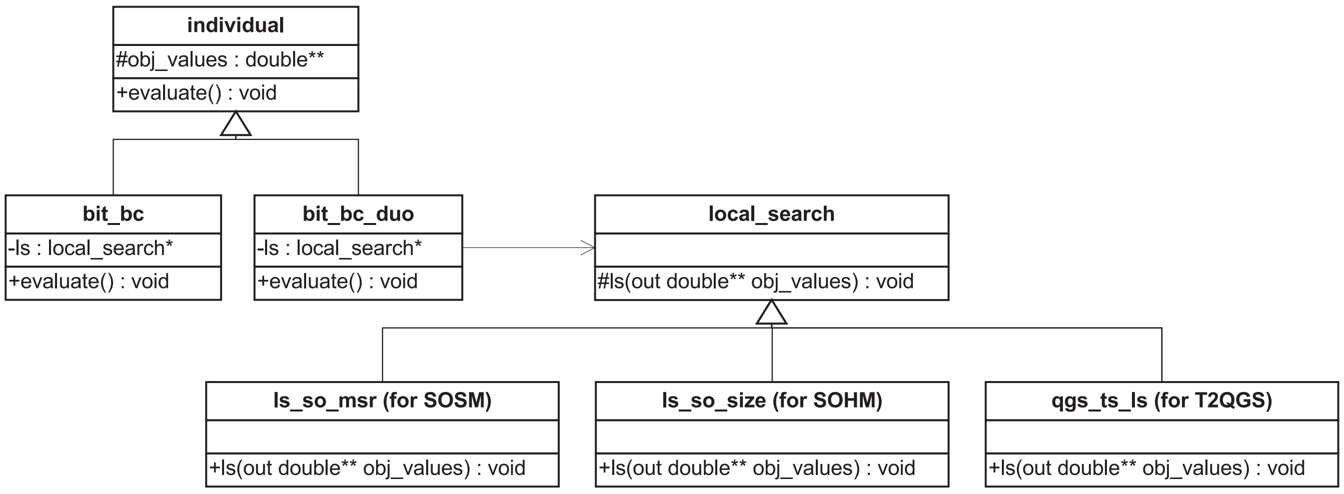


Figure 4.3: UML diagram to show the interdependence of the two classes `individual` and `local_search` (and their descendants) in the PISA framework. This class structure allows for a modular handling of the local search strategies (STRATEGY design pattern [10]).

tions `alg_1`, `alg_2`, and `alg_3` newly merged into the following two functions:

- `alg_1_sizeconstraint` implements a single and multiple node deletion algorithm for multiple data sets. Each dataset is considered independently. The single node deletion removes the node whose exclusion leads to the largest increase in homogeneity, i.e. to the largest decrease of the MSR. The mechanism for multiple node deletion should help to improve the performance; under the assumption that some threshold measure (that needs to be recalculated in every iteration) remains valid for the removal of several nodes, we can remove nodes based on this approximate measure. To avoid too much inaccuracy, we allow this procedure only for cluster sizes above some threshold value (`row_threshold`, defined in `moop_rank_ls_param.txt`; typical value: 1'000).

- `alg_3_sizeconstraint` implements a single node addition algorithm. First it checks all genes, whether its addition would increase the MSR for any of the participating files. The gene is added if and only if this is not the case. Then the same procedure is repeated for the chips.

We are not going to present results to this problem. Firstly, it served as a quick way to get familiar with PISA framework and yielded preliminary results only. Secondly, this local search is computationally still very expensive and prevents us from extensive statistical analyses unless the code is optimized. And thirdly, this work is dedicated to the query gene approach that we want to investigate in detail. However, we documented the SOHM local search for completeness.

4.3.2 T2QGS Local Search Strategy

Compared to the SOHM algorithm in the previous chapter, the T2QGS is a fundamentally different clustering approach. This is also reflected on the

Algorithm 1: Deterministic repair function.

```
$no\_genes$ : number of genes in cluster
$min\_no\_genes$ : constraint on minimum number of genes
$r\_mean$ : row mean value over all data sets
**if** $no\_genes < min\_no\_genes$ **then**
    $pi \leftarrow$ sort $r\_mean$ in ascending order and store the resulting
    permutation of indices
    add the first $no\_genes - min\_no\_genes$ genes of $pi$ that are not in
    the cluster to the cluster
**end**
```

implementation side, where we developed the T2QGS local search from scratch instead of just adopting some functions to fit the new requirements as we did beforehand; there are 1600+ lines of source code dedicated to the local search only.

A key function is `qgs_t2_ls::ls` which is called from inside the evaluation function of each individual (Figure 4.3). Its task is both to assure the feasibility of the resulting solution and to assign all objective values. It operates in four stages:

1. **Repair individual to ensure feasibility.** The EA may generate individuals that violate a constraint. For instance, this is the case if the number of experiments $|C|$ of the bicluster is smaller than the minimum number of experiments the constraint `min_chips_constraint` (defined in `qgs_t2_ls_param.txt`) demands. Of course, the neatest way would be to have an encoding that makes infeasible solutions impossible. Since this does not hold for our representation, we can apply a repair function that transforms an infeasible individual into a feasible one. We now have at least two possibilities to achieve this, in particular in a deterministic or in a stochastic fashion. EAs are inherently randomized algorithms and on one hand one could consider it appropriate to randomize the repair function and with this the evaluation function. On the other hand one does not feel certain about the outcome of this practice because two identical (and in our case infeasible) individuals will not necessarily evaluate to the same objective values anymore. There are algorithms that can handle randomized evaluation functions, but for our purpose we employ a deterministic scheme. For details see Algorithm 1.

2. **Remove genes far away from the query gene.** Removing genes from the bicluster should be done such that the genes closest to the query gene remain in the bicluster. We first employ a multiple node deletion algorithm that achieves an exponential reduction of the number of genes in the cluster over the iterations, Algorithm 2. To this end, it calculates the overall mean distance $\bar{d}$ of all genes $\in G$ over the selected experiments C of all files. Then those genes with a mean distance above $\bar{d}$ are removed. If the number of genes in the cluster drops below some threshold value t_r, the algorithm switches to single node deletion. The threshold t_r is usually set to twice the minimum number of genes constraint, because if t_r would be chosen below that value, we might produce an infeasible solution that would not be repaired anymore. Above we would (probably needless) suffer from additional CPU time. The

Algorithm 2: Single and multiple node deletion algorithm.

```
$no\_genes$ : number of genes in cluster
$min\_no\_genes$ : constraint on minimum number of genes
while $no\_genes > min\_no\_genes$ do
    foreach $gene \in G$ do
        $row\_avg[gene] \leftarrow$ calculate average over all data files of $gene$
    end
    $overall\_avg \leftarrow$ mean of $row\_avg$
    foreach $gene \in G$ do
        if $row\_avg[gene] > overall\_avg$ then
            remove $gene$ from cluster
            $no\_genes \leftarrow no\_genes - 1$
        end
        if $no\_genes \leq 2 \cdot min\_no\_genes$ then
            break
        end
    end
end
```

overtaking single node deletion algorithm removes the farthermost gene until we reach the minimum number of genes constraint.

3. **Add closest genes.** For the gene addition, we have at least two options to accomplish the task:

   (a) Add gene if its distance to the query gene is among the closest $q\%$ of all genes $\in G$. q is a parameter named `ls_add_genes_q`, defined in `qgs_t2_ls_param.txt` and typically set to 0.5 in our case. From this strategy it is not clear how it respects every objective value f_i: it tries to add as many genes as possible ($\rightarrow$ favoring size objective) but it does not add genes that are too far away from the query gene regarding G ($\rightarrow$ restricting maximum dissimilarity). The other option distinguishes the influence of the f_i more clearly.

   (b) Make explicit use of the objective values by applying some dominance-based measure. This would make sense insofar that it emulates the selection strategy imposed from the selector (i.e. IBEA). Unfortunately, this is a population-based measure that requires population wide information we do not have available in an individual's local search. This obstacle cannot be avoided, but for instance, we could alternatively apply a local dominance based search strategy: add a gene if it improves all objectives f_i. Indeed, such dominators potentially exist, since our distance measures are based on *mean values* where adding below-average gene decreases the its value. The size objective f_{n+1} will always be decreased by the addition of a gene[1].

However, since the latter option implies consequences that are difficult to

[1]Strictly speaking, this holds only for moderately small clusters with a number of genes below the maximum genes constraint, see below.

Algorithm 3: Gene addition algorithm.

```
$no\_genes$ : number of genes in cluster
$r\_mean$ : row mean value over all data sets
foreach $gene \in G$ do
    $row\_avg[gene] \leftarrow$ calculate average over all data files of $gene$
end
$overall\_avg \leftarrow$ mean of $row\_avg$
set threshold
$t \leftarrow min(row\_avg[]) + q \cdot (max(row\_avg[]) - min(row\_avg[]))$
foreach $gene$ in $G$ do
    if $row\_avg[gene] < t$ then
        add gene to the cluster
    end
end
```

estimate, we decided to implemented option (a). A more detailed description of the addition algorithm is shown in Algorithm 3

4. **Assign appropriate objective values.** In memetic algorithms in general and also in ours, the assignment of objective values is often left to the local search. There are actually two perceptions of evolution regarding what happens over an individual's lifetime, named after their initiators: Lamarck and Baldwin. In the theory of Lamarck, individuals genetically inherit skills acquired by their parents, where Baldwin stated that an individual can only pass on genetic information that it inherited itself (with some attenuation). This carries over to our case as follows: if we want to simulate Lamarckian evolution, an individual is assigned the point (local optimum) that emerged from the local search. In contrast, Baldwinian evolution attributes an individual its *fitness value* from the solution found by the local search but the individual undergoes *no genetical changes* after the local search. Two such runs will be discussed in the section on results.
The computation of the fitness values is described in Algorithm 4. As given in the problem description, all objectives are to be *minimized*. We have two types of fitnesses to assign: homogeneity and size.

   - We base the calculation of the homogeneity objective on the actual distance measure. The bicluster $\{G, C\}$ induces an objective value f_d for file d,

     $$f_d = \sum_{g \in G} \text{dist}\,(g, C_d)^2$$

     where C_d denotes the set of columns of the bicluster that belong to file d. We have a total of n files.
   - The size objective calculation involves at the same time a constraint handling technique to penalize large clusters. Basically, a bicluster is assigned a size objective value of

     $$f_{n+1}(\sigma) = 1/\sigma \tag{4.1}$$

where σ indicates its size. If we don't intervene, the algorithm will make use of the whole domain of cluster sizes ranging from the minimum number of genes constraint up to the total number of genes. From a biological point of view this is not desirable because we are not expecting to find biclusters of that size at all. Therefore we introduce a soft constraint on the cluster size. Earlier, we mentioned two techniques for constraint handling; here we apply a third one and penalize the cluster size if it exceeds some given value. There is a variety of possibilities to achieve this; anyhow there are two conditions one could reckon as meaningful:

(a) f_{n+1} should be C^0-continuous. Since EAs are essentially zero-order algorithms, one can argue that any higher order continuity will probably not influence the performance.

(b) $f_{n+1}(M+k) \overset{!}{=} f_{n+1}(1) \overset{(4.1)}{=} 1$, where M is the maximum number of genes constraint. The kth larger gene than M ($k \in \mathbb{N}$) is demanded to have the same value as the smallest feasible cluster.

We saw in a preliminary test that an exponential penalty appears to be too severe compared to a polynomial penalty. Using a (continuous) ansatz with a polynomial of oder p, $f_{n+1}(x) = a + b\,x^p$, one finds that in general

$$a = \frac{M^{p+1} - (k+M)^p}{M\,(M^p - (k+M)^p)}, \quad b = \frac{M-1}{M\,((k+M)^p - M^p)}. \tag{4.2}$$

There is no good argument for higher order polynomials and that is why we set $p = 1$. This reduces Eq. (4.2) to

$$a = \frac{-M^2 + M + k}{k\,M}, \quad b = \frac{M-1}{k\,M}.$$

From this we define the size objective as

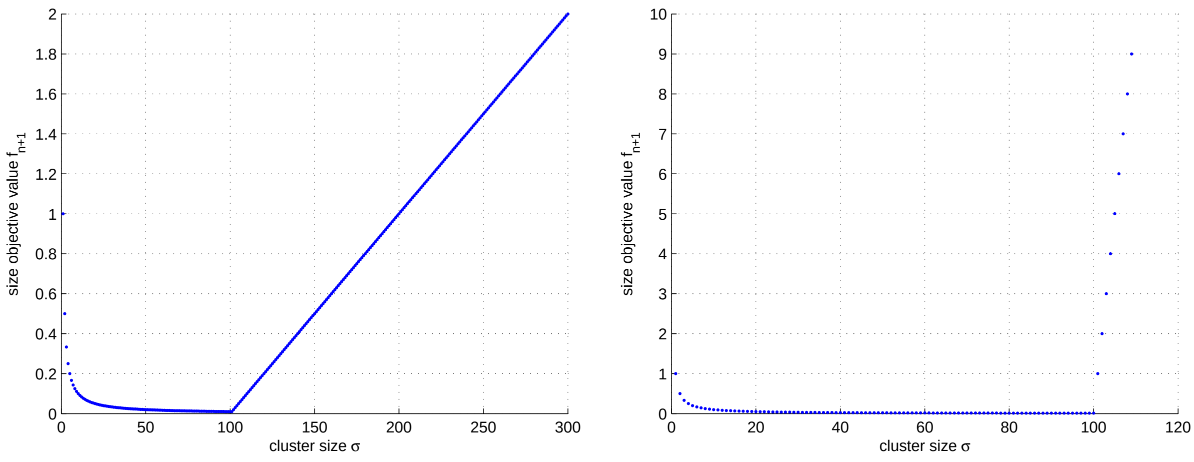
$$f_{n+1}(\sigma) = \begin{cases} 1/\sigma & \text{if } m \le \sigma \le M \\ \frac{-M^2+M+k}{k\,M} + \frac{M-1}{k\,M}\,\sigma & \text{if } \sigma > M \end{cases}, \tag{4.3}$$

with $\sigma \in \mathbb{N}$ and $m \in \mathbb{N}$ minimum number of genes constraint.

Note that this function is invertible only for $k = 1$ (and strictly speaking $f_{n+1} \neq 1$), which can be may be valuable for understanding and comparing the algorithm with versions where the size objective is not an optimization target. Figure 4.4 visualizes Eq. (4.3).

We were also concerned with the normalization of objective values which would facilitate the comparison between the different objectives. At this point, we suffer from a problem that we have already faced in the context of gene addition; to normalize the objective values to a certain range, e.g. [0,1], one would need population wide information that is not available at the level of the local search. Neither we are allowed to scale each objective vector, which would be normalization within every individual instead of normalization over the generation.

One way out would be to run one generation and then store the minimum



4.4a Parameters: M = k = 100.

4.4b Parameters: M = 100, k = 1.

Figure 4.4: Size objective function with penalty for soft constraint violations.

values for each objective globally such that this can be used by the subsequent fitness calculations for normalization. Since doing so destroys the possibility of comparing between the different runs and furthermore did not improve the performance of the algorithm, we decided not to apply a scaling to the objectives. Typical orders of magnitude for the values of homogeneity objectives are 10^{-2} where for the size objective values this is approximately 10^{-3}.

4.4 Equal-bit flip Mutation Operator

The PISA framework implements two standard mutation strategies: independent bit flip and single bit flip. The former strategy flips every gene and every chip bit with some fixed probability. Single bit flip performs one single bit flip drawing from an uniform distribution. Both of these strategies potentially disrupt solutions because the individuals we are interested in tend to have much more zeros, $\mathcal{O}(10'000)$, than ones, $\mathcal{O}(10)$, where a method that balances ones and zeros in expectation will impose a bias towards much larger clusters. Therefore we introduce a new bit flip strategy that flips as many ones to zeros as zeros to ones. It chooses a random number in $[0,$ `total_genes`$-1]$ and seeks for the next zero in order to flip it to 1 (wrap around). It counts how many times a one was flipped. Then it applies the same procedure to flip the same number of ones to zero. Details are given in Algorithm 5. For this mutation operator, the number of bits in a cluster is only changed by cross-over. By 'uniform initialization' we ensure that there are not only individuals with all bits set to one.

4.5 Restricting the Local Search

The local search is known to be quite strong in finding local optima. This can be a cause for a low diversity of the population but also prevent the algorithm from exploring certain solutions. Therefore we implemented a restriction

Algorithm 4: Objective assignment.

```
$dist\_arr$ : array of distances to the query gene for each file and each
gene (total $n$ files)
$max\_no\_genes$ : maximum number of genes (soft constraint)
$obj\_val$ : array of objective values ($n + 1$ objectives)
foreach $file$ of the problem do
    $obj\_val[file] \leftarrow 0$
    foreach $gene \in G$ do
        $obj\_val[file] \leftarrow obj\_val[file] + (dist\_arr[file][gene])^2$
    end
end
if use size objective then
    $size \leftarrow no\_genes \cdot no\_chips$
    if $no\_genes \leq max\_no\_genes$ then
        $obj\_val[n + 1] \leftarrow 1/size$
    else if $penalty\_type ==$ exponential then
        $obj\_val[n + 1] \leftarrow$ apply exponential penalty
    else if $penalty\_type ==$ linear then
        $obj\_val[n + 1] \leftarrow$ apply linear penalty
    end
end
```

for the local search: it does not allow the algorithm to add or remove more than `max_add_del_genes` genes from a solution. This rule only applies if the number of genes in the cluster is below some threshold value, typically `max_no_genes` defined in Algorithm 4. If this threshold was omitted it would take many generations for large clusters to shrink to a feasible size. The user may choose via the parameter file `qgs_t2_ls_param.txt` whether this restriction strategy should be applied or not.

4.6 Implementation Concept

We conclude this chapter with a an important implementation concept we followed: we tried to never dismiss a former functionality for the sake of an extension or even new one. So at many points throughout the code you will find branches where new functionality is implemented but yet is not in conflict with already existing code. The decisions at these branches are controlled by the parameter files and for this reason, at each level (PISA, ebica, and xxx_ls) there is a parameter file.

Algorithm 5: Equal bit flip mutation operator. It is applied to the gene and chip bit string independently. Here we show the case for genes.

```
$length$ : length of input bit string
$no\_genes$ : number of genes in cluster
$mut\_prob$ : probability for a bit mutation
**for** $i = 1$ to $N$ **do**
    **if** double_rand([0,1[) $< mut\_prob$ **then**
        **if** $no\_genes \neq 0$ **then**
            $b \leftarrow$ int_rand([0, $total\_genes$])
            flip the next 1 to a 0 beginning at position $b$
            $no\_genes \leftarrow no\_genes - 1$
        **end**
        $b \leftarrow$ int_rand([0, $total\_genes - 1$])
        flip the next 0 to a 1 beginning at position $b$
    **end**
**end**
```

Chapter 5

Results

This section summarizes the outcome of our investigations. To this end we document the experimental setup and define therein a reference case we can refer to later. The actual results are split in two sections: the first compares the output of a SOSM run versus a T2QGS run for a sanity check and then details the performance of the T2QGS algorithm by comparing several 2D size constrained and a 3D size optimized runs. The parameters we thereby varied are presumed to behave sensitive. In a second and shorter section, we present observations made during the development of the T2QGS algorithm. These preliminary results are meant to indicate tendencies only, since they lack statistical validation. In a last section we turn to convergence and consider to this end the development of the objective values over the generations.

5.1 Experimental Setup

5.1.1 Data sets

For the runs we made use of two real and one artificial input data set. All simulations were performed on gene expression data generated with Affymetrix GeneChips from *Arabidopsis thaliana*, a small plant. A more detailed description is given in Table 5.1.

5.1.2 Reference Run

We want to define a reference run and specify its parameters in detail. Doing so allows us to give only the deltas for other runs compared to this datum run. Table 5.2 shows the settings.

5.1.3 Performance

The run time for T2QGS runs strongly depends on several parameters. Of course, increasing the population size, the number of generations etc. makes the run more expensive. Apart from this, the evolution type (Lamarck/Baldwin), the distance measure and clustering/biclustering have a strong impact on the run time. The reference run with 150 generations (instead of 100) took about 288 mins[1] to

[1]All simulations were run on a Intel Xeon 3.06 GHz CPU with 2 GB RAM.

Name	*Description*
COLD_FULL	A series of experiments investigated the response of *Arabidopsis thaliana* to different kinds of stresses. The related data was provided by the AtGenExpress consortium and consists of 8 time series with 6 time points each. Therefrom we chose 2 time series to form the COLD_FULL input data set that contains 22'746 genes and 12 experiments. This data set represents a case where the expression values are well comparable across the different time courses; the experimental setup was identical for all different kind of stresses, all measurements were performed by the same laboratory using the same microarray technology, the expression values were normalized and log ratios were calculated using measurements from an untreated control plant.
HETERO_FULL	This input data set also consists of two data sets, where the first is the same as for the COLD_FULL input data set. The second has been taken from a measurement series that is much more diverse than those of the first data set; it includes different types of treatments such as heat stress, infection with Pseudomonas syringae, and measurements of diurnal changes. All expression values were normalized. In contrast to the first data set absolute expression values are used. Our input data set has the dimensions 2 $\times$ 22'746 genes $\times$ 6 experiments.
RAND_N	Again, RAND_N consists of two data sets with 22'746 genes and 2 $\times$ 6 experiments that were sampled from a standard Normal distribution $\mathcal{N}(0, 1)$. We chose this distribution since it is the distribution with maximum entropy and thus the least information content. This input data set served mostly for the study of distance measures.

Table 5.1: Input data sets in use.

Parameter	*Value*
Data set	HETERO_FULL
Query gene (specific for data set)	23
Size objective	yes
Min. number of genes constraint	10
Max. number of genes constraint, k	200, 100
Min. number of experiments	6 (all), 11 (all)
Mutation probability	0.1
Number of generations	100
Evolution type	baldwinian
Tournament size	2
Soft constraint type	linear
Distance measure	ranked MSR
Rank scaling	yes
Constrain local search	no

Table 5.2: Parameters of the Reference Run.

complete where a 2D reference run (constraint on the size) yielded a run time of about 212 mins.

Regarding the T2QGS algorithm, there is a highly promising optimization potential: if the user agrees to be content with clustering instead of biclustering, the internal distance tables, holding the distance from every gene to the query gene for every file, become only data dependent. They have to be calculated only once for the whole run. For biclustering, we have to recalculate the distance table for every generation and each individual, since in- or excluding experiments changes the distance; using look-up tables is in the case of biclustering infeasible, because there would be $2^{\sum_i |C_i|}$ many tables to keep in the memory (where $\approx$ 3 MB each). If the user insists on biclustering, we can apply a less powerful but yet worthwhile optimization: provided that we use any distance measure that is the mean of a quantity q,

$$\bar{q} = \frac{1}{N} \sum_{i=1}^{N} q_i,$$

and want to, say, remove one element from q_k from the average to receive the new mean $\bar{p}$, we can achieve this by

$$\bar{p} = \frac{1}{N-1} (N\,\bar{q} - q_k).$$

5.2 Extensive Investigations

We have introduced a few new concepts and a lot of related parameters. Since we are confronted with a combinatorially large number of possible constellations we selected a small subset of parameters that are potentially sensitive with respect to the outcome of a run. This set encloses the boolean whether or not to include the size objective, the query gene, the minimum number of genes constraint and the

random generator seed. In Section 5.2.2 we present extended results on the impact of these parameters.
During this project we have also set up single simulations that indicated tendencies we did not pursue further in statistical evaluable runs, because we concentrated on the abovementioned investigation of parameters. Nevertheless those results may be of use and we will summarize them in Section 5.3.
But before we start out to study the performance of the algorithm on the selected parameter set, we show that the T2QGS algorithm is able to recover an OPSM the SOSM algorithm already found. Further we define a reference run that serves as datum for the other simulations.

5.2.1 SOSM vs. T2QGS

Here we want to perform a basic sanity check of the T2QGS algorithm. To this end we show that the multi-objective T2QGS approach is able to recover the same perfect[2] bicluster of size 32 genes $\times$ 12 experiments the SOSM algorithm finds in the COLD_FULL data set.
First we ran the SOSM algorithm with the homogeneity constraint δ set to zero. This forces the SOSM algorithm to search for OPSMs only. The number of experiments constraint is set to the number of experiments, such that we are concerned with clustering. The largest bicluster of this kind that can be found has the abovementioned size of 32$\times$12; see Figure 5.1 for the expression profiles of both datasets.

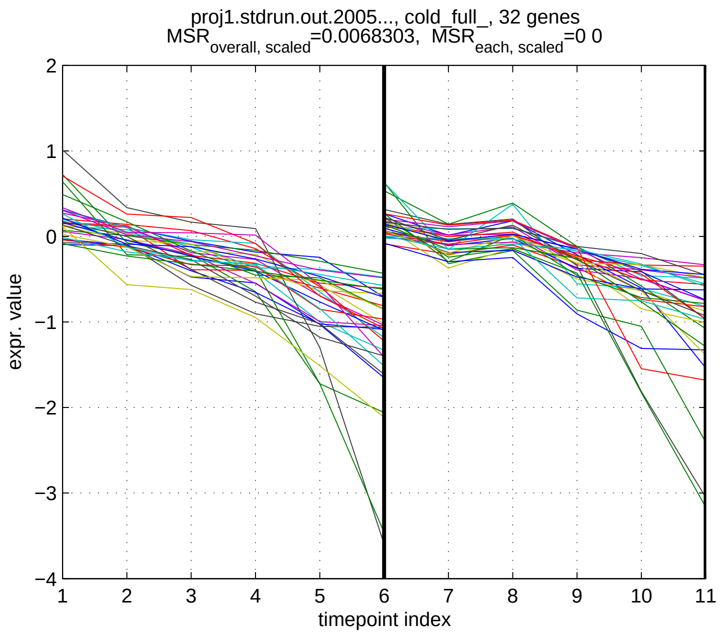


Figure 5.1: The largest cluster that was revealed by both the SOSM algorithm and the T2QGS approach. It contains 32 genes and 12 experiments and is an OPSM.

[2]With 'perfect' we refer to a bicluster that is an OPSM and therefore has a vanishing MSR score.

Second, given the above run, we chose a gene from the 32×12 cluster and input it to the T2QGS algorithm as the query gene. There is no size objective for this comparison and we have only the two data files from the COLD_FULL data set, each of them acting as one objective. The solutions were constrained to a minimum of 10 genes and all experiments in both data sets.
For three out of three different query genes, all of them being members of the 32×12 cluster, T2QGS returned the same cluster that we found earlier with the SOSM approach depicted in Figure 5.1. The algorithm seems to work as expected.

5.2.2 Including the Size Objective

Here we compare the performance of the T2QGS algorithm in a case where we include the cluster size as an additional, competing objective versus a case where we constrain the EA to a minimum number of genes. We are clustering on the HETERO_FULL input data set containing 22'746 genes and 2 × 6 experiments. Thus, we have

- two homogeneity objectives f_1, f_2 and
- one size objective f_3

to minimize. The result of the reference run (3-objective optimization) is given in Figure 5.2 in the objective space. The front has quite a narrow shape which can be ascribed to scaling. We neither made use of IBEA's option to scale the objectives nor did we introduce a scaling on the variator side. Since scaling is something we imposed on the problem and that has not changed over all runs, we will not run into troubles because of that. However, this can be illustrated by multiplying the third objective by 50 such that it is transformed into the same absolute order of magnitude as the first and the second. Figure 5.3 shows that the shape of the front has become wider.
In order to compare the 2D and 3D results (without and with the size objective, respectively), we proceed as follows. Since the size objective f_3 of a 3D run (namely the reference run) is directly related to the size $\sigma = |G| \cdot |C|$ of the cluster, $f_3 = \frac{1}{\sigma}$, one can treat the iso-objective line $f_3 = \xi$, use $|G| = \frac{1}{\xi\,|C|}$ and set up a 2D run with a minimum number of genes constraint set to |G|. Since we want to compare the two cases in two dimensions, we select all individuals from the 3D run with $f_3 \leq \xi$ and filter them to obtain non-dominated solutions only. After filtering out all dominated solution from the 2D run as well, we can summarize the two fronts in one single plot. This conservative strategy has been applied to a variety of cases as shown in Figures 5.5–5.8 for a first query gene and in Figures 5.9–5.12 for the other query gene. We carried out simulations for the query genes 23 and 1313 in HETERO_FULL, neither of them belonging to a known cluster and for each of them with and without optimizing the size. Those runs that do not use the size as an objective have four different size constraints (15, 37, 58 and 66 genes). Every of these combinations has been run with five random number generator seeds to make sure we're not looking into an artifact caused by the very seed. The same applies to those cases including the size as an objective.
Summarizing, there are $2 \cdot 4 \cdot 5 + 2 \cdot 5 = 50$ runs in total to investigate, Figure 5.4 tries to give a rough overview. The runs are identical to the reference run (Section 5.1.2) except for the four abovementioned varied parameters plus the number of

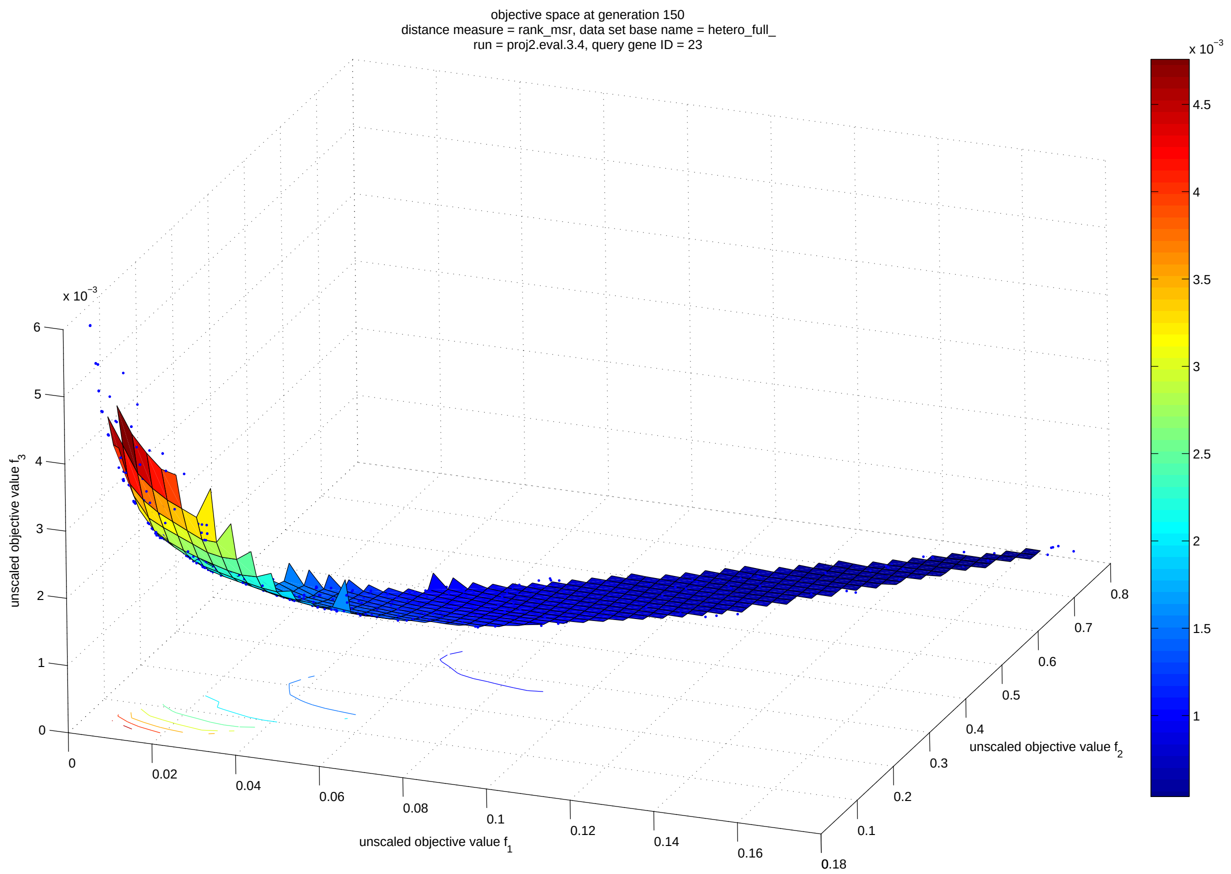


Figure 5.2: An approximation of the Pareto front of a 3-objective optimization problem (parameters conforming with the reference run).

generations parameter which we increased from 100 to 150.

The 2D and 3D runs both have the same number of individuals in their populations. When we leave all parameters unchanged and move from 2 to 3 objectives, the algorithm has to distribute the individuals in an additional dimension. This drastically affects the resolution, even if we add only one single dimension and besides, it makes it more difficult for the algorithm to explore/exploit the search landscape. This finding reminds of a known phenomenon called *curse of dimensionality* which in general describes the observation that the number of points needed to sample a space equidistantly grows exponentially in the dimension of the space. For our case this translates into a massive increase in the number of non-dominated solutions (if drawn randomly). Note that if we wanted to work against this problem, we had to increase the number of individuals exponentially which would yield an exponential increase in CPU time. Observing Table 5.3 shows the expected tendency. We ran one simulation (for query gene 23 and seed no. 4, cf. lowest plot in Figure 5.8) with a population size of 1000 individuals, which visibly improves the number of solutions that are non-dominated (and therefore displayed), see Figure 5.13.

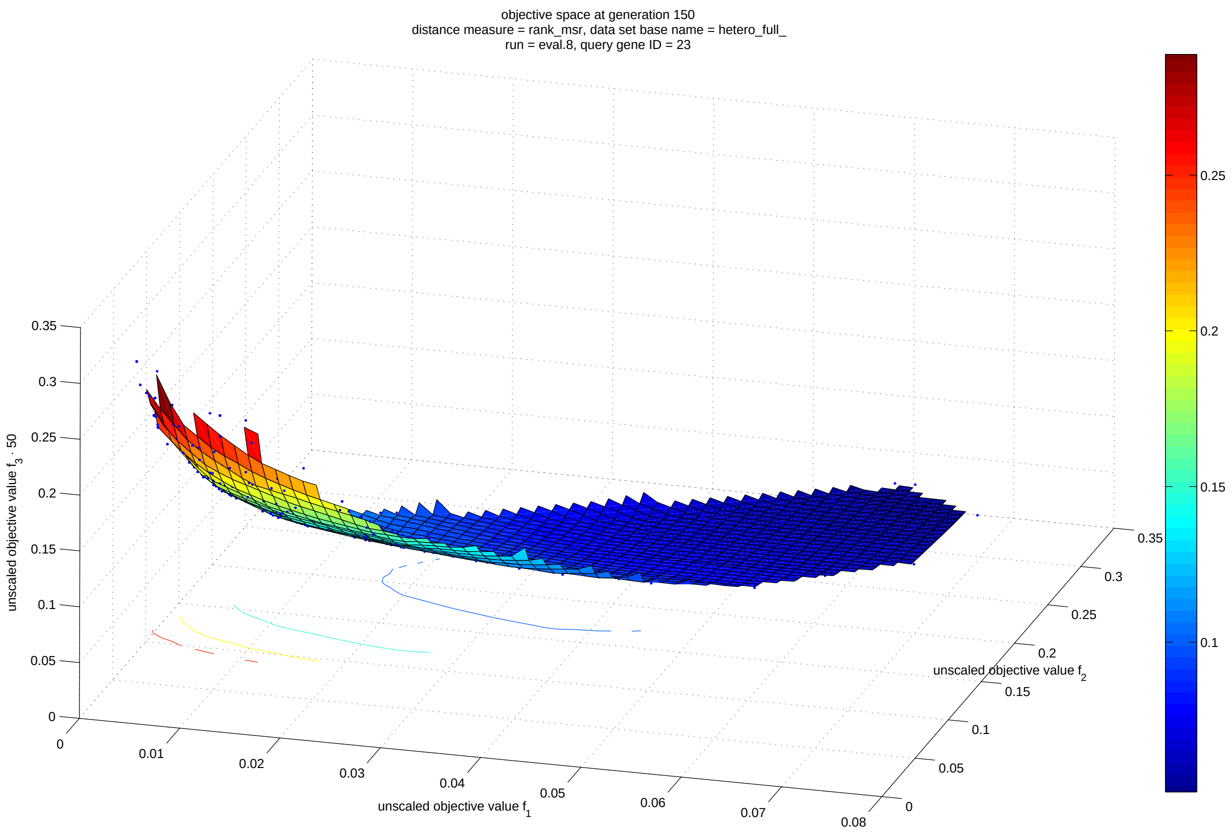


Figure 5.3: The identical problem as in Figure 5.2 but with rescaled size objective $f_3 \leftarrow 50\, f_3$.

Another interesting point is the size distribution of the individuals in the last population for we can learn something about how the algorithm distributes the solutions over the front and simultaneously check whether the soft constraint accomplishes its task. We have upper bounded the cluster size to 200 genes by a soft constraint on f_3. Figure 5.14 shows the mean, minimum and maximum number of genes over 5 runs of the two 3D runs from above. The number of genes in the clusters are almost uniformly distributed in the interval [15, 60] for the first (Figure 5.14a) and [15, 80] for the second query gene (Figure 5.14b). However, this is a range for the size where biologically relevant clusters are expected to occur.

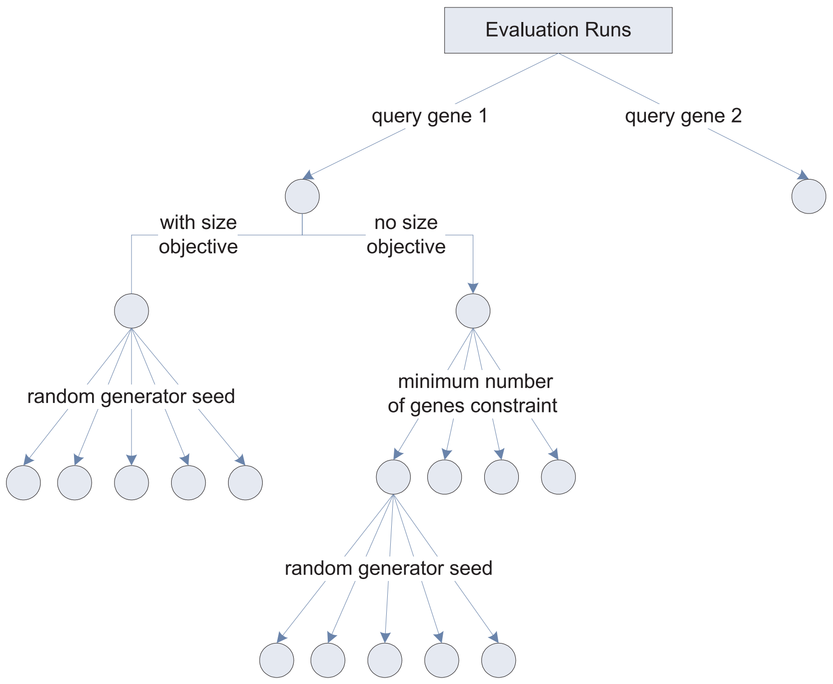


Figure 5.4: Total 50 runs that investigate the impact of including the as size constraint in the optimization. The tree is symmetrical, reoccurring branches are shown only once.

	eval.1.1.{0–4}	eval.1.2.{0–4}	eval.1.3.{0–4}	eval.1.4.{0–4}
min. # genes	15	37	58	66
2D: # non-dom. sol.	218.4 (27.5)	162.4 (23.0)	40 (14.5)	59.8 (37.8)
3D: # non-dom. sol	367.4 (5.4)			

Table 5.3: Number of non-dominated solutions for the different 2D/3D comparison runs. Regarding the population size of 400 individuals, the impact of the additional dimension is heavy. All runs used query gene 23. We gave the means over 5 runs and in brackets the sample's standard deviation.

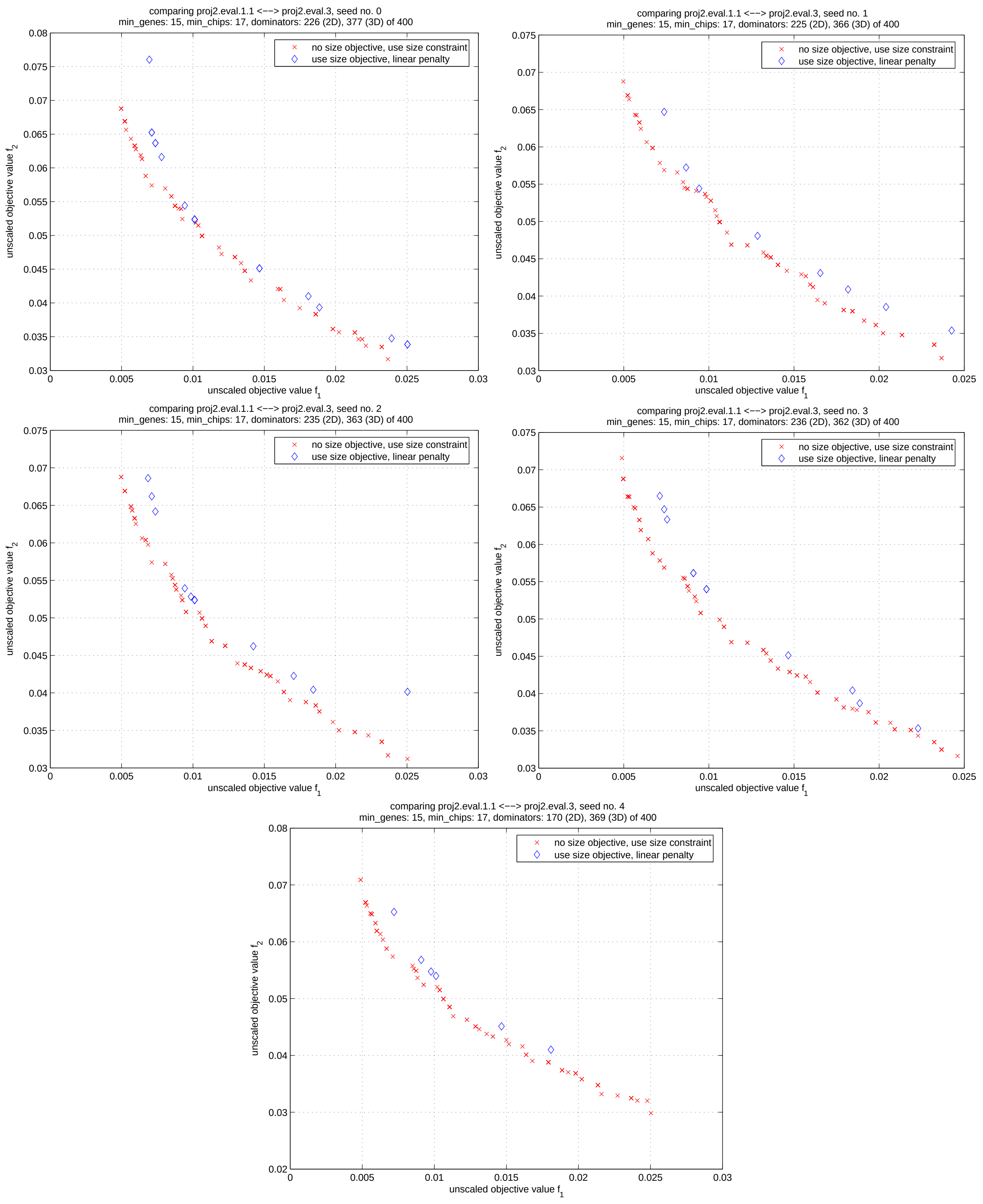


Figure 5.5: Impact of size as objective: query gene 23, comparing clusters of size 15.

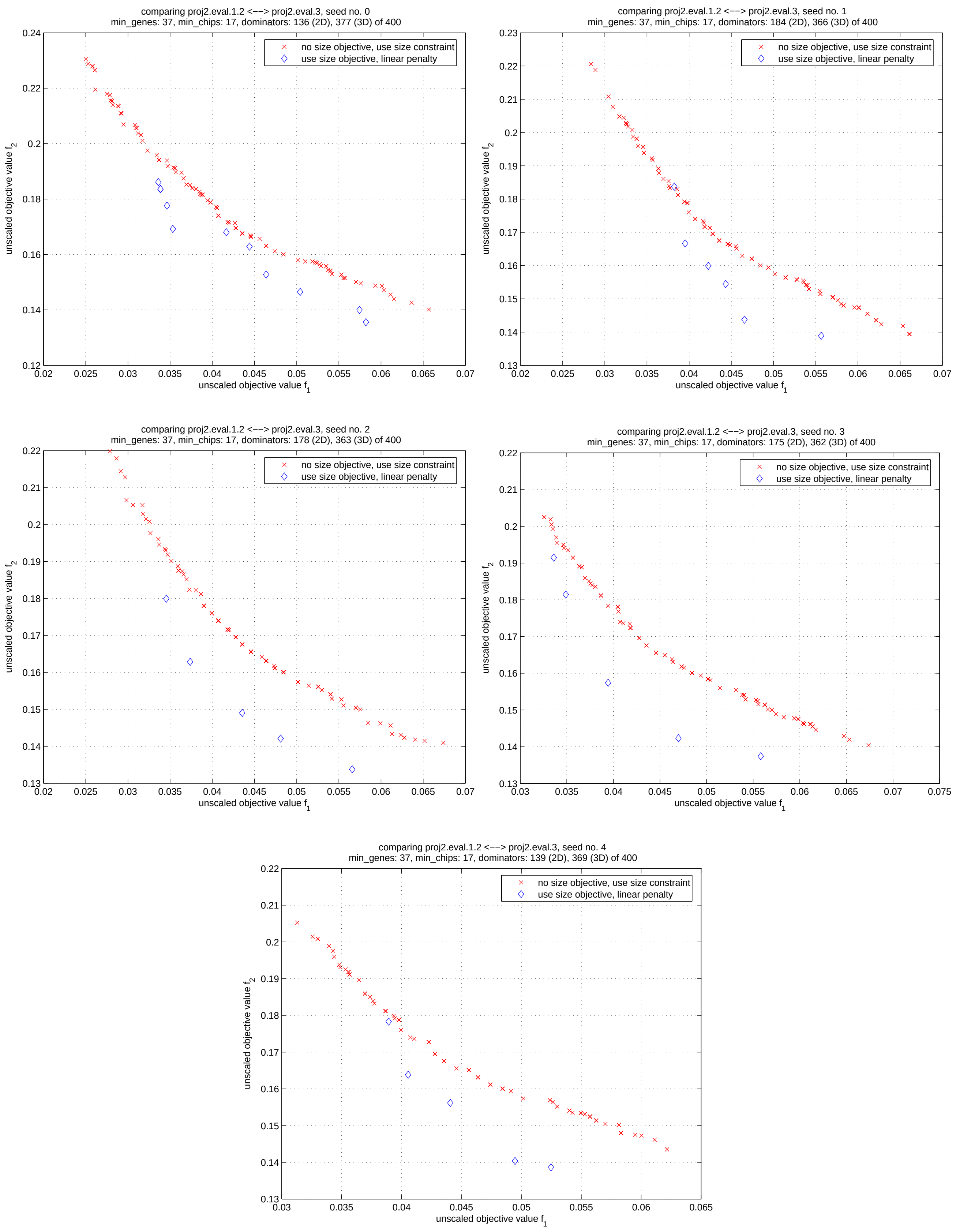


Figure 5.6: Impact of size as objective: query gene 23, comparing clusters of size 37.

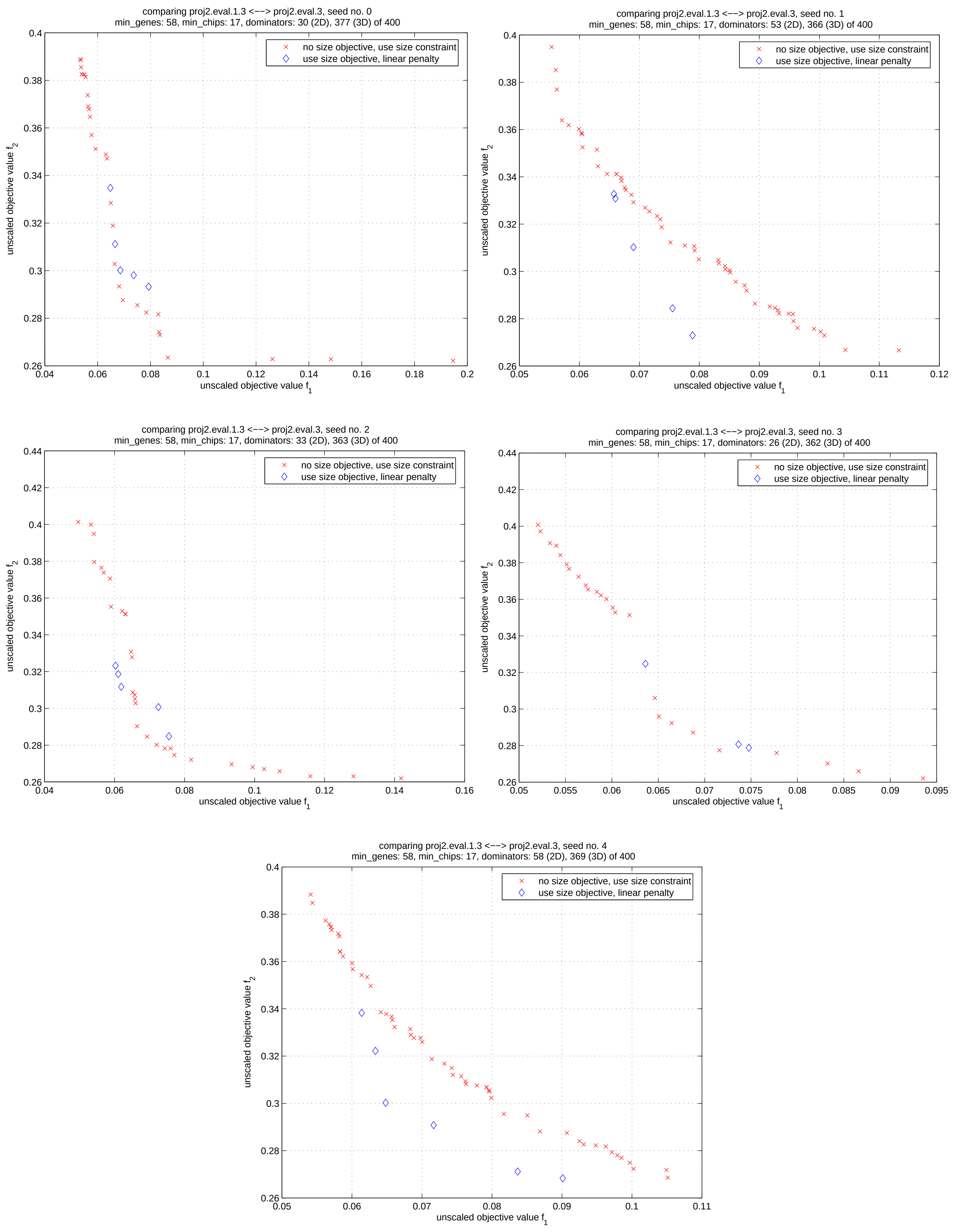


Figure 5.7: Impact of size as objective: query gene 23, comparing clusters of size 58.

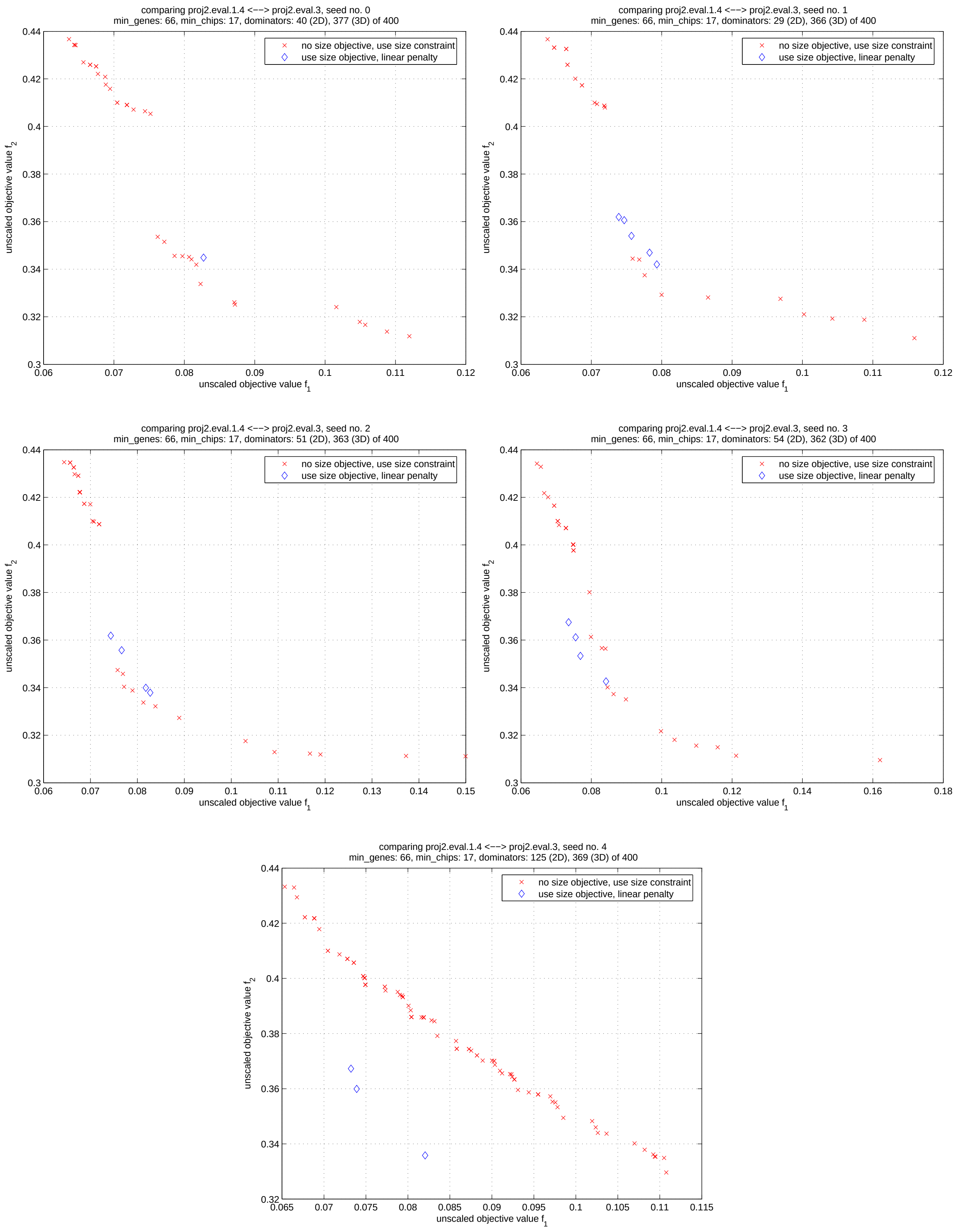


Figure 5.8: Impact of size as objective: query gene 23, comparing clusters of size 66.

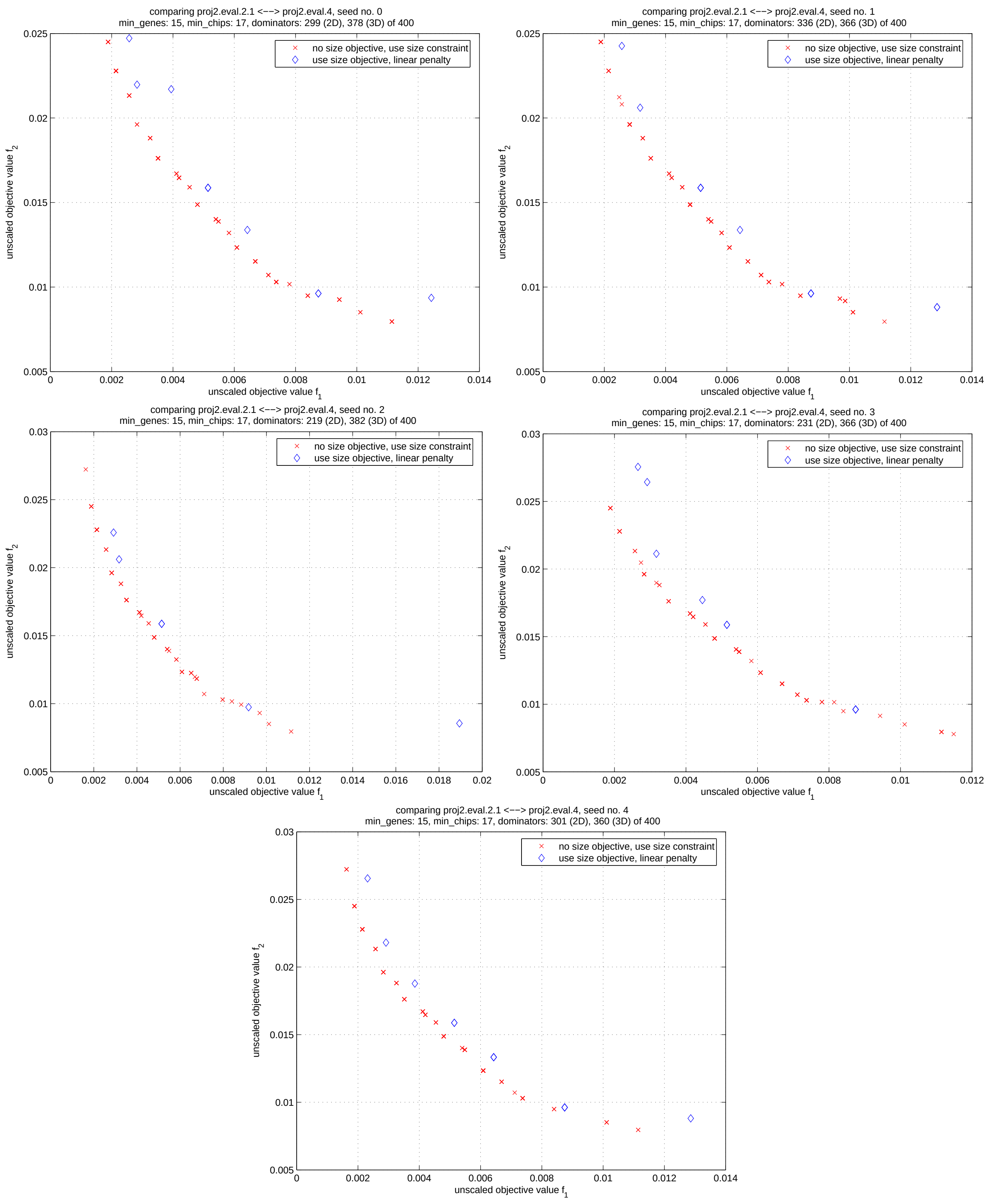


Figure 5.9: Impact of size as objective: query gene 1313, comparing clusters of size 15.

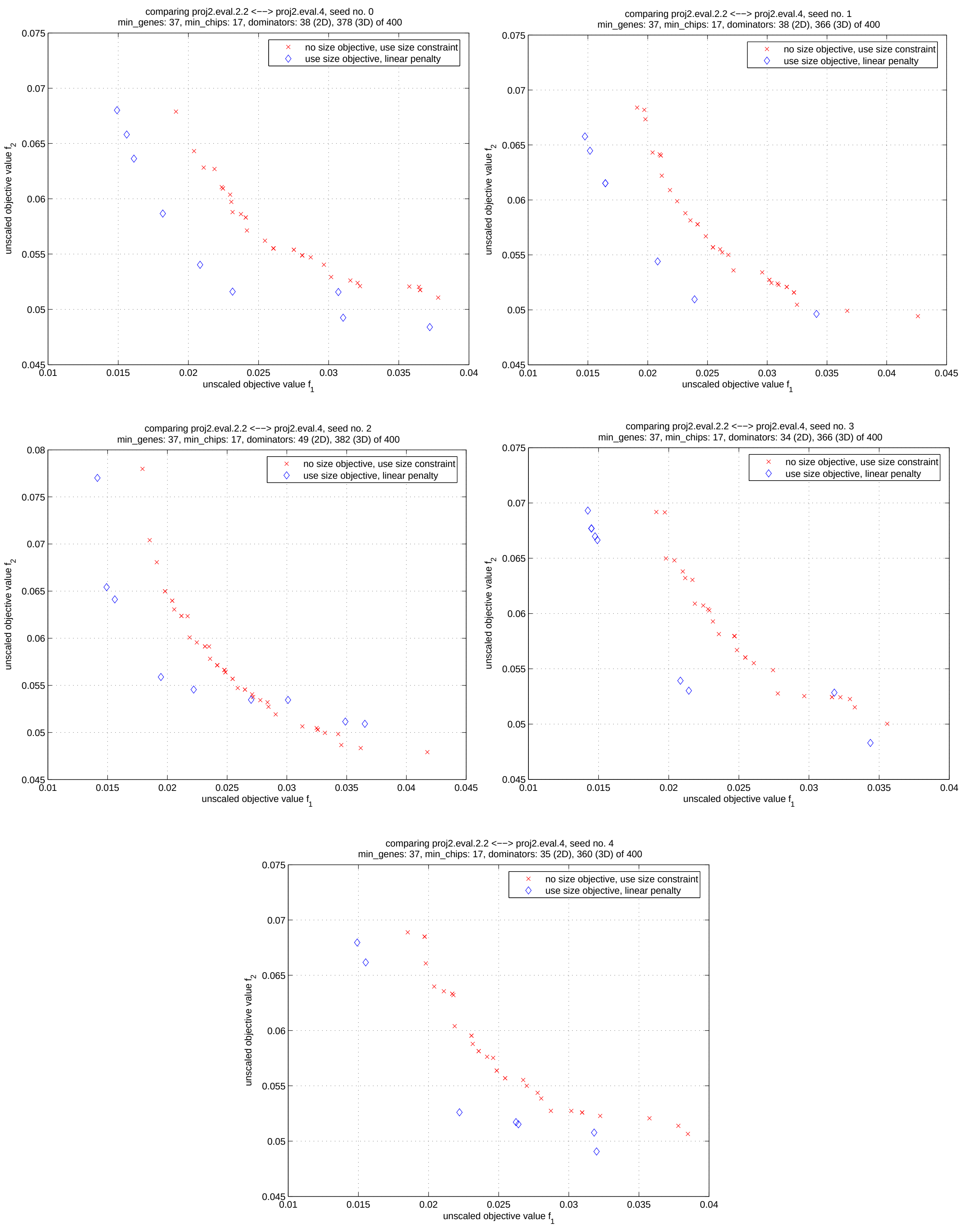
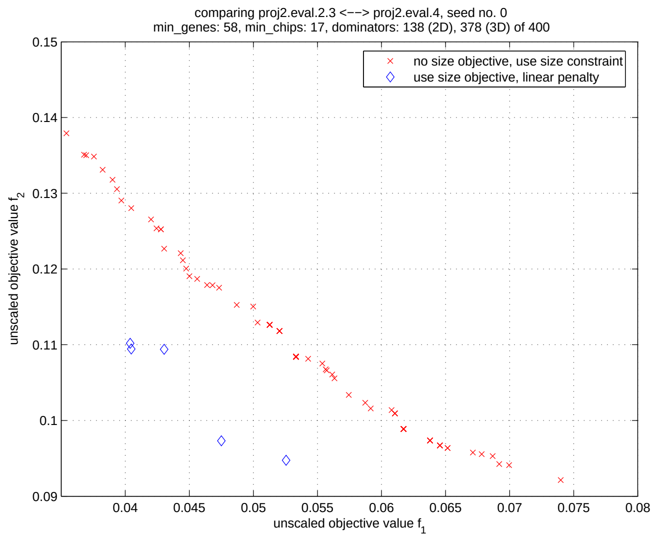
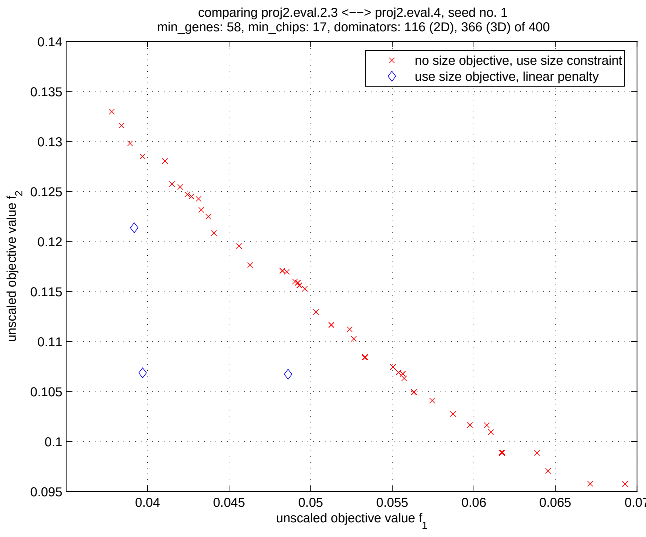
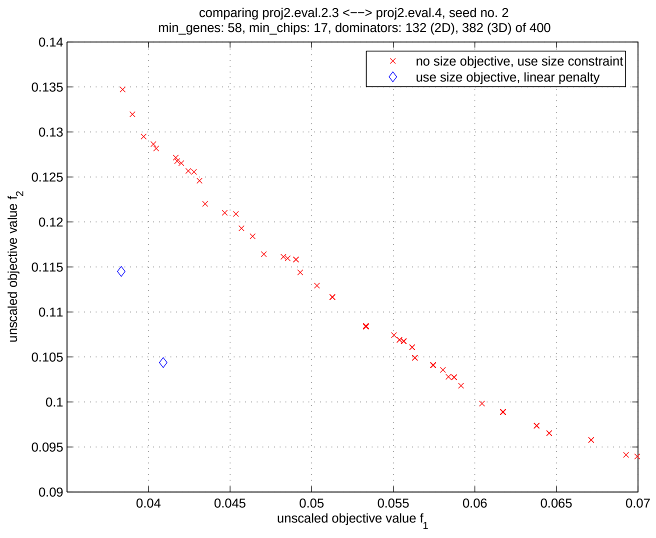
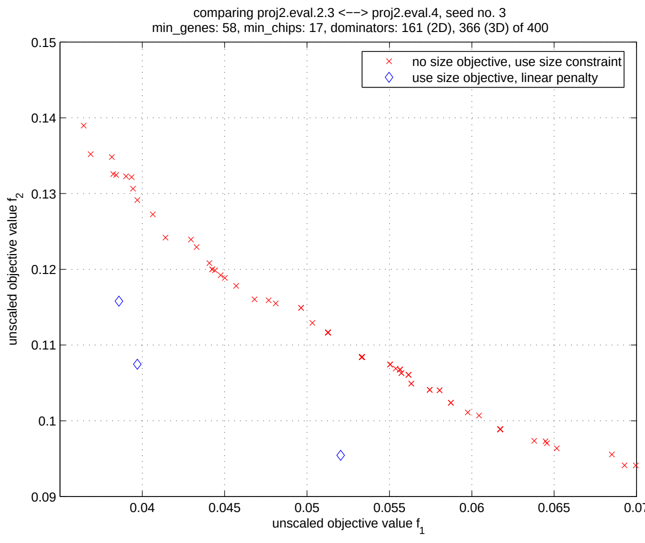


Figure 5.10: Impact of size as objective: query gene 23, comparing clusters of size 37.









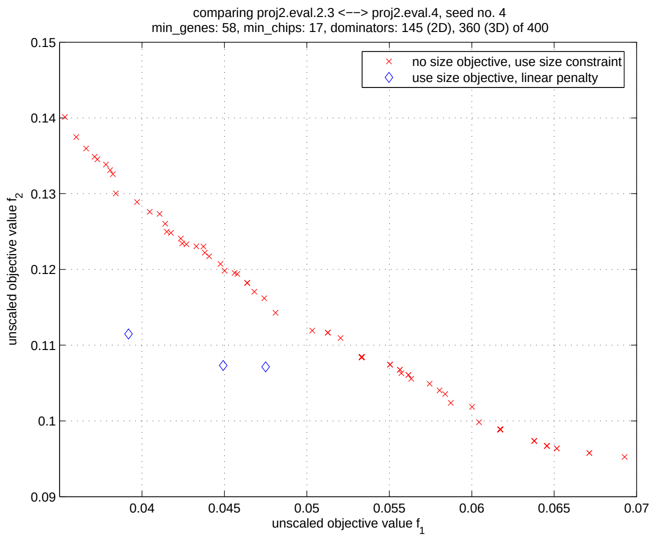
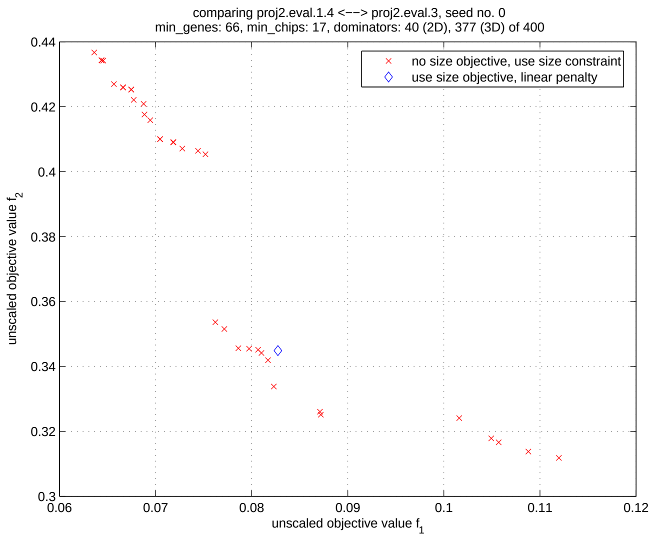
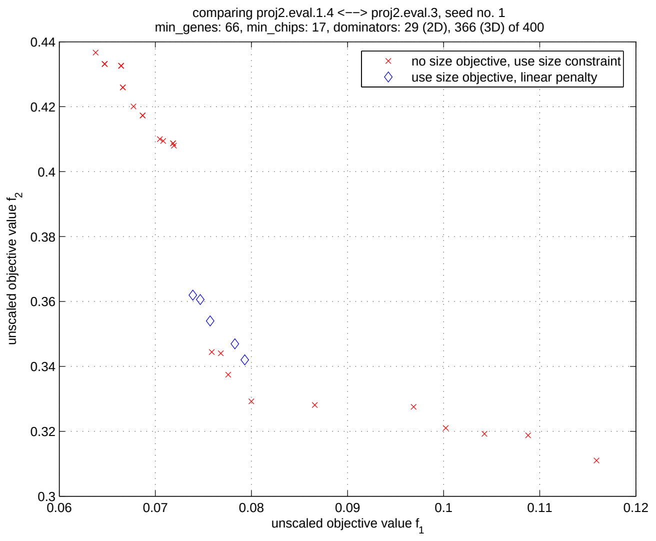
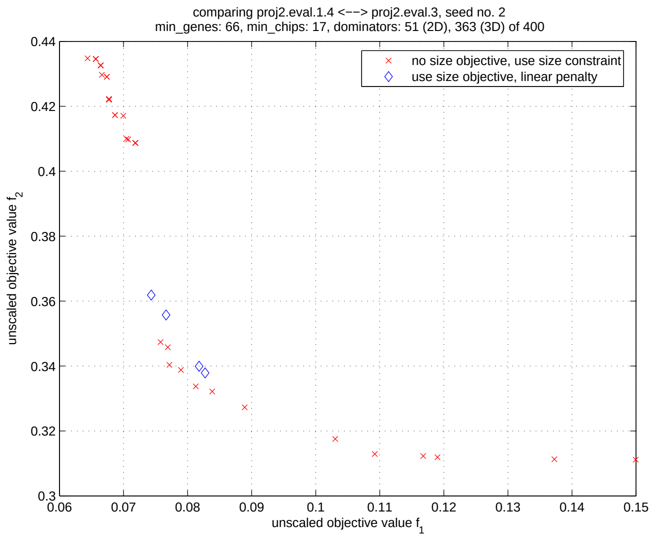
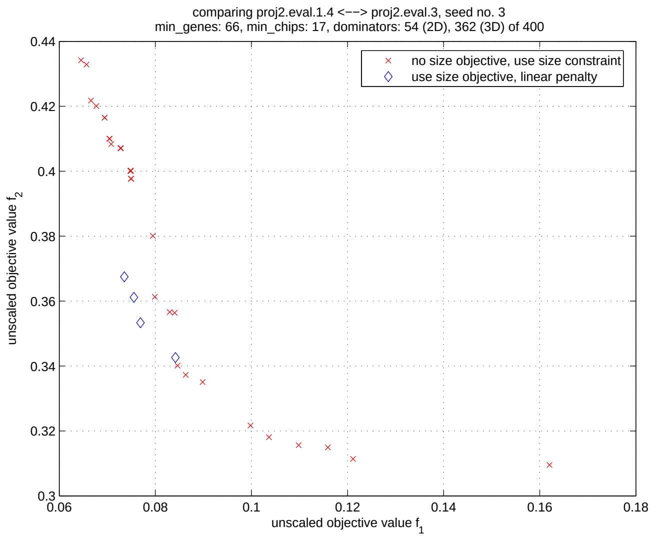


Figure 5.11: Impact of size as objective: query gene 1313, comparing clusters of size 58.









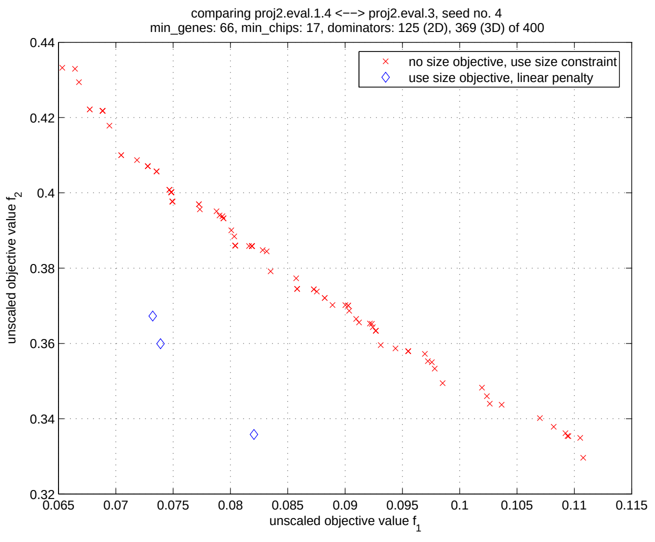


Figure 5.12: Impact of size as objective: query gene 1313, comparing clusters of size 66.

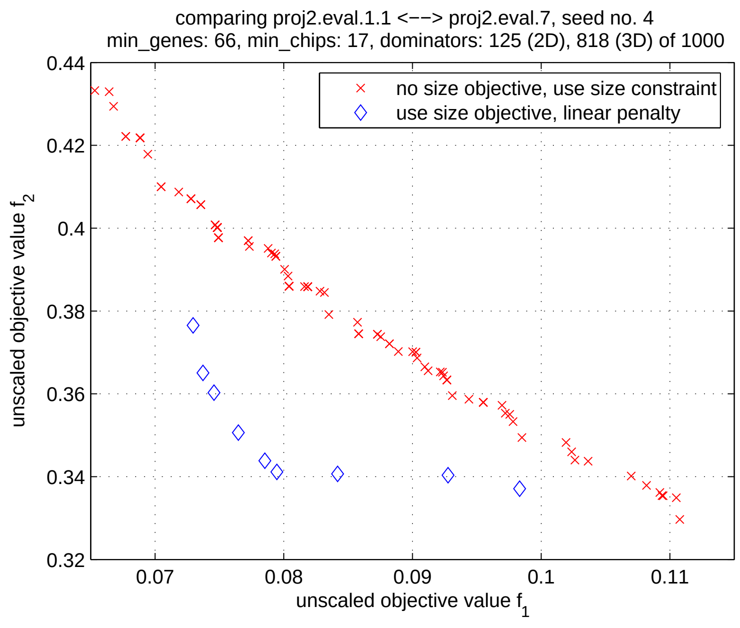
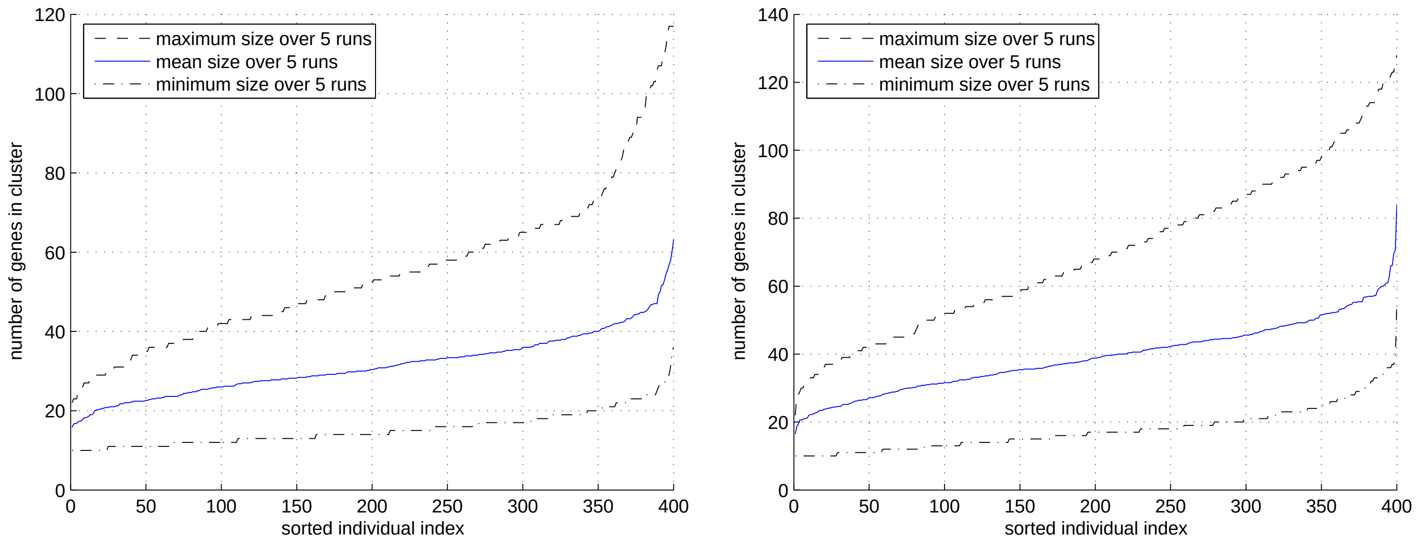


Figure 5.13: The same run as in Figure 5.8 (seed 4), but with 1000 instead of 400 individuals. Obviously, the number of points of the (projected) 3D front has increased.



5.14a Query gene 23.

5.14b Query gene 1313.

Figure 5.14: Distribution of the number of genes in two size-optimized runs.

5.3 Observable Tendencies

Here we address observations made during the development of the T2QGS algorithm and note tendencies on the basis on a few preliminary results. Since the results in this section are not scientifically validated by statistical means, the results below are mainly given to indicate directions for further investigations and/or for completeness.

5.3.1 Baldwinian vs. Lamarckian Evolution

Baldwinian evolution increases the runtime of the algorithm substantially but brings about a more diverse final population regarding the sizes of the individuals. Since the T2QGS approach is anyway restricted by giving a query gene, we are not concerned with the diversity. Figure 5.15 compares two runs, one with Baldwinian and one with Lamarckian evolution. The final Lamarckian population contains almost exclusively the same individual.

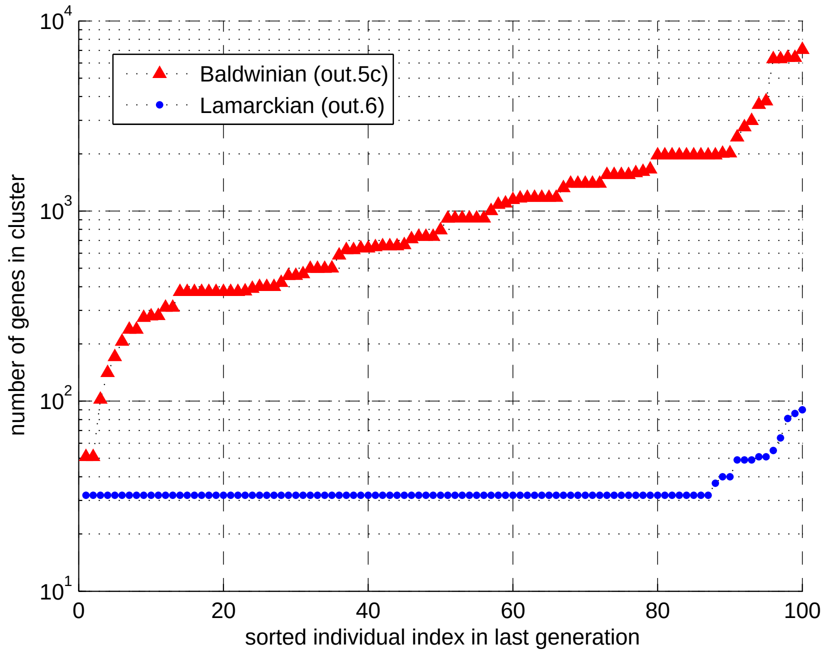


Figure 5.15: Diversity of cluster sizes: Baldwinian vs. Lamarckian evolution.

5.3.2 Equal-bit-flip mutation operator

Section 4.4 already introduced the new mutation operator. We set up a run that does not optimize the size but is directly derived from the reference run, differing only in in the mutation strategy. From this single run we don't have reason to believe that the mutation strategy has a fundamental impact on the solution; Figure 5.16 compares the final populations of the two runs.

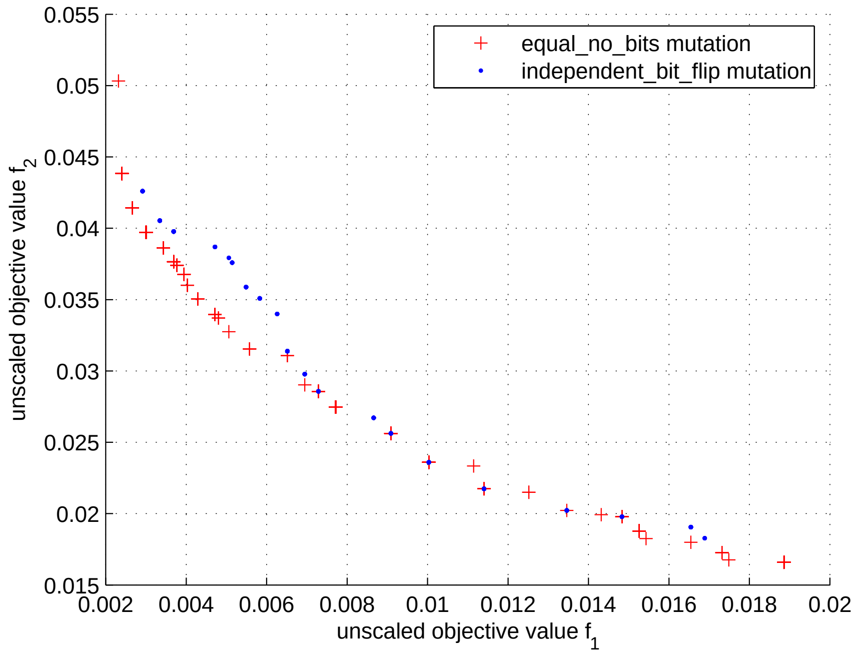


Figure 5.16: The impact of the new mutation strategy on the front appears negligible.

Besides, we noticed an increase in run time for the equal number of bits strategy compared to independent bit flip of about 5%.

5.3.3 Quality parameter k in gene addition procedure

We are mainly interested how the maximum size soft constraint parameter influences the related size objective. The test runs differ from the reference run only in the way too large clusters are penalized, here we used an exponential instead of linear penalty function. However, this does not affect the meaning of the parameter.
From Figure 5.17 we mainly observe that admitting a larger size constraint entails the advantage of a smaller size objective value. At the same time, however, we observe a larger fraction of individuals that received a size penalty. The seemingly plausible tendency here is that with an increasing size constraint we sacrifice feasible individuals.

5.3.4 Full Biclustering

From the SOSM problem it is known that if the constraint on the number of experiments is lowered, the resulting bicluster will most likely follow this lower limit closely. In other words it seeks to maximize the bicluster's size by adding more genes rather than including more experiments.
Since the T2QGS problem is fundamentally different from the SOSM problem, it is natural to ask what happens to the number of experiments included in the bicluster, if we set the minimum genes constraint for every file to 1. Obviously, and

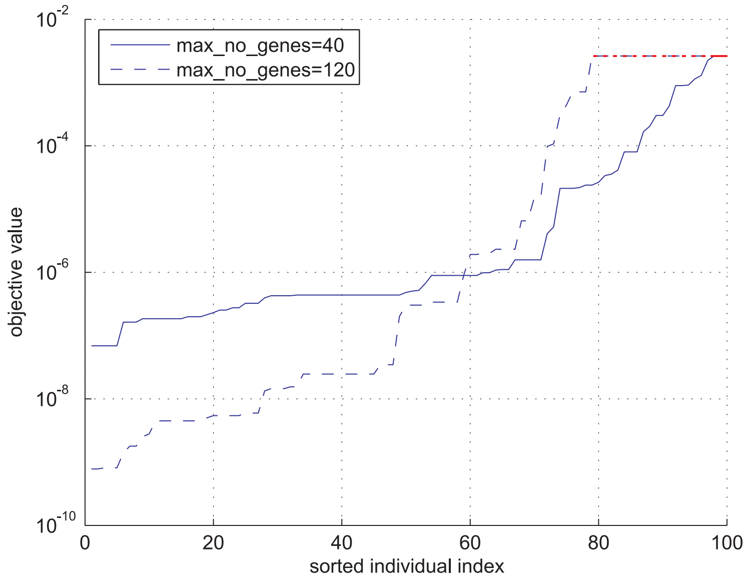


Figure 5.17: Influence of the maximum number of genes constraint. The segments in red (dashed and solid) indicate that the related individuals received an infeasible objective value.

different from the SOSM solution, the T2QGS algorithm decided to include in 25 of 100 individuals *all* of the experiments, and for the remaining 75 individuals all except one. The sizes of the biclusters vary from 33 to 86 genes, see Figure 5.18. The plot also shows on the secondary axes (red) the number of experiments included in the bicluster. Interestingly, from a certain number of genes, it sees an advantage in excluding an experiment. This is the point where the competing objectives meet: additional genes in the cluster must be bought by excluding experiments due to the homogeneity objectives.

For this run we clustered on the COLD_FULL input data set that is known to contain quite similar data and thus facilitates large, homogeneous clusters. For instance, the largest cluster of the above investigation with 86 genes has still very low MSR values (Figure 5.19): $1.5 \cdot 10^{-31}$ ($\ll$ machine eps) on the first and $6.5 \cdot 10^{-3}$ on the second data set. Biologically meaningful clusters may have MSR values up to $5 \cdot 10^{-2}$ [4], depending on the data set.

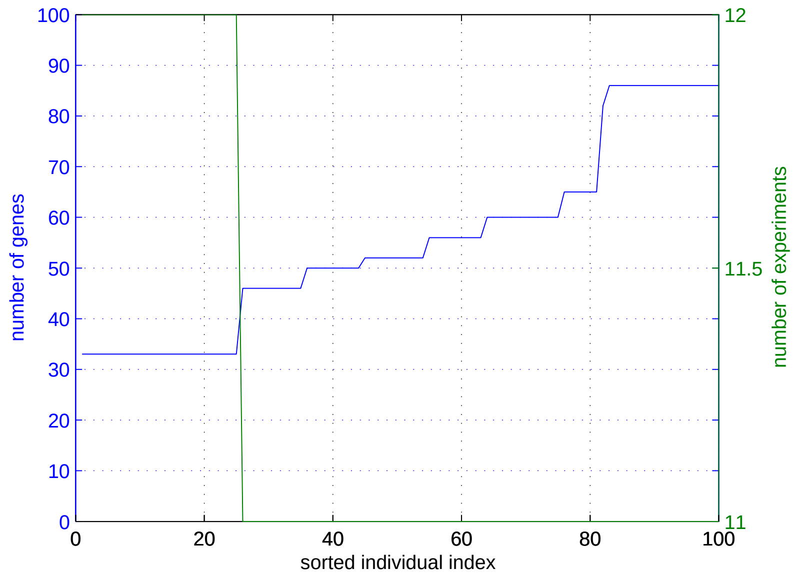


Figure 5.18: Size distribution of the last generation for a run differing from the reference run in the mutation strategy (independent bit flip) and max. number of genes constraint set to 40 (k = 80).

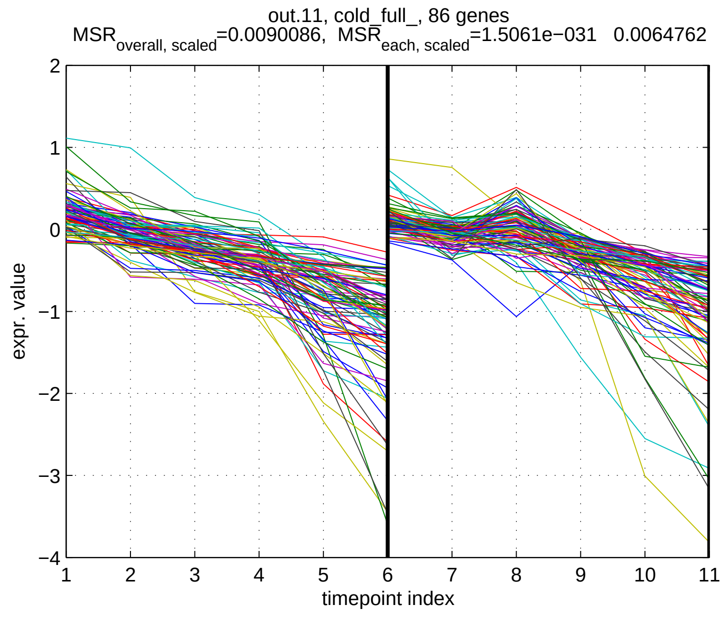


Figure 5.19: A large bicluster in COLD_FULL with 86 genes.

5.4 Objectives over Generations

Plotting the objective values over the generations gives an idea whether convergence takes place or whether there are unwanted fluctuations in the solution. In Figure 5.20 we depict the development of the three objectives of the reference run over the generations. The values remain stable after some initial transient oscillations. Obviously, f_1 and f_2 *increase* after a few generations. This is most likely an artifact from multi-objective optimization rather than intrinsic to this very problem, since we could observe this behavior in almost all of the runs.

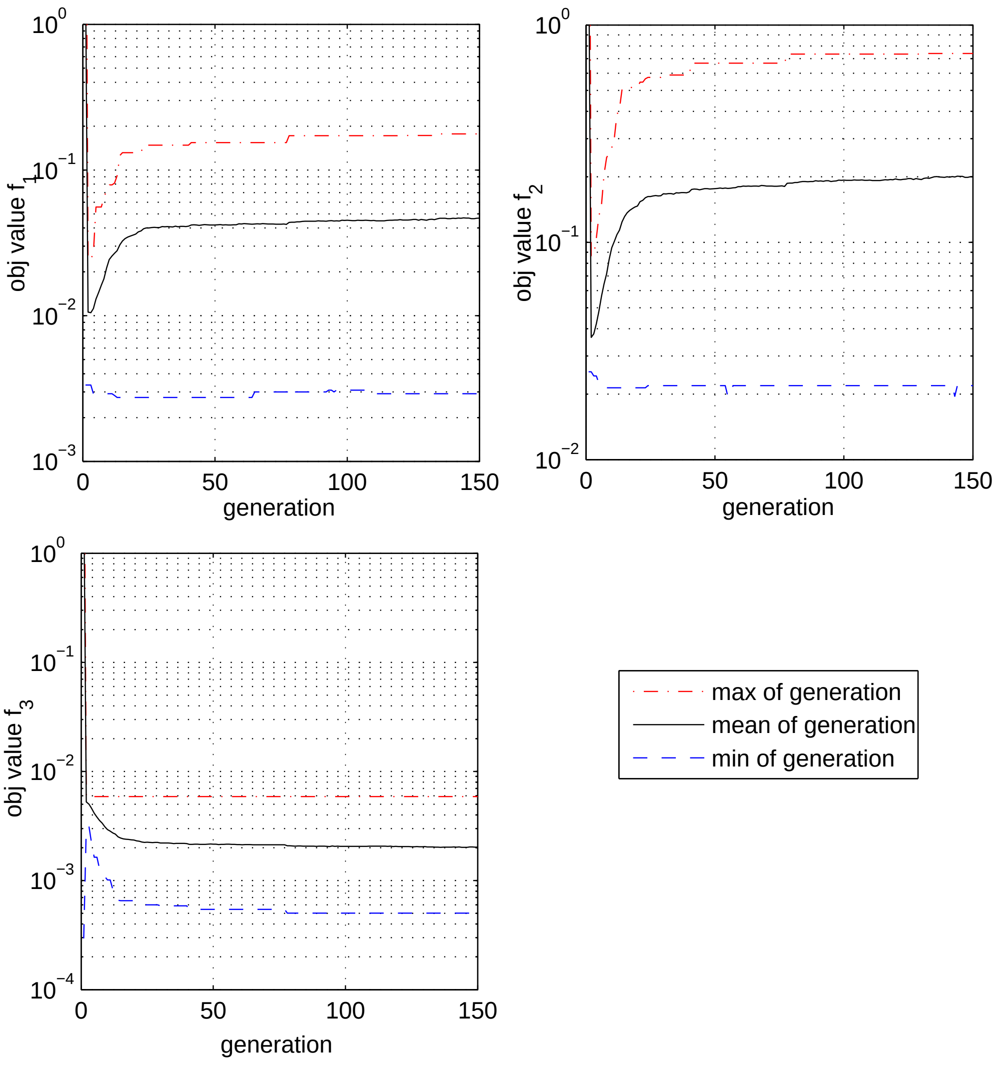


Figure 5.20: Residuals over 150 generations of the reference run (except that we have 150 generations here, instead of 100).

Chapter 6

Conclusion and Outlook

In our query gene clustering problem, we are looking for a subset of genes of the expression matrix (a cluster), such that the expression profiles yield maximum homogeneity over multiple data sets. In terms of an EA this translates to multiple criterions that we need to satisfy and we tackle this by assigning each objective the homogeneity of a data set. However, we saw that there exists a clear trade-off between these objectives. For a basic validation of the newly developed T2QGS algorithm[1], we showed that it is able to recover an already known cluster that has been discovered by a single objective evolutionary algorithm. This validation algorithm tries to maximize the size of the cluster constrained to a minimum homogeneity value.

Usually, one is interested in maximizing the size of a cluster and therefore we included the size as a further objective in the search. Again, we found that this objective is clearly competing with the other objectives. For a comparison, we set up a run with two data files and another run with the same data files but with the size as a third, additional objective. In most cases we observed *objectively* better solutions for the 3D runs, even though the number of individuals was held constant. Including the size as a further objective forces the algorithm to explore an additional dimension which decreases the resolution. To compensate this loss one has to increase the population size, which directly affects the computational costs. However, in view of the still moderately small run time of about 2 hours[2] for a standard run, one usually accepts the additional costs in favor of the substantial improvement of the cluster quality.
Interestingly, the cases where the 3D run is superior to the 2D run reside in a size range that is potentially meaningful from a biological point of view[3]. For small clusters of about 17 genes and for an equal number of individuals, the results from the 3D run were slightly inferior to those of the 2D run. For clusters with 37, 58 and 66 genes we found a substantial improvement in the clustering outcome for almost all of the cases. We compared a total of 40 cases that affirm these assertions.

[1]The T2QGS algorithm is implemented in an existing framework tailored to multi-objective optimization with an evolutionary algorithm (PISA).
[2]All simulations were run on a Intel Xeon 3.06 GHz CPU with 2 GB RAM.
[3]The crucial quantity for biological relevance is the homogeneity of the cluster, rather than the absolute number of genes. However, for a given input data set, one can *roughly* state a number of genes that corresponds to the critical cluster homogeneity.

From previous work it is known that the abovementioned validation algorithm tries to maximize the size of a *bi*cluster by including more genes rather than more experiments. Lowering the constraint on the minimum number of experiments entails that the algorithm will most likely follow this lower bound closely. However, this does not hold for the T2QGS algorithm: setting the minimum constraint on the experiments to 1 for each file yielded biclusters containing nearly or even all of the experiments. This shows that the two problems fundamentally differ in this aspect.

Furthermore, we investigated Lamarckian and Baldwinian evolution. The latter uses the result of the local search only for the assignment of the objective values where the former immediately updates the individual with the result. Thus, Baldwinian evolution is in general computationally more expensive than Lamarckian. Nevertheless we showed that in the case of a query gene search the diversity of the cluster size is not significantly affected and that the cheaper Lamarckian evolution shows a sufficient performance for the query gene search problem.

There are many interesting questions in this context that are worth being addressed in further work. In the list below, we name a few of them.

- **Normalization.** For many selection strategies it is desirable to work with objective values that are of a comparable order of magnitude. Normalization has to take place on the population level and is already provided by IBEA for example. We urgently suggest to explore its effect.

- **Multiple Data files.** In this thesis we presented cases with two data files where each was assigned an objective. If we do so for more and more data sets, we will find a rapidly increasing number of non-dominated solutions in the population. Thus we are looking for a method that can handle this problem. A possible approach could use dimension reduction techniques from statistics, e.g. PCA/ICA (principal/independent component analysis), EPP (exploratory projection pursuit), or LLE (locally linear embedding).

- **Multiple Query Genes.** Our algorithm can handle exactly one query gene; straightforward, an extension would allow to have more than one query gene. If we assign each of them an objective, we face the same problem as above. For instance, this could be bypassed by taking the mean of the distances from all genes in the cluster and using this quantity as a new distance measure. Alternatively, one could apply the gene recommender algorithm from Owen et al as a local search strategy. A third possibility consists in solving a one single-query gene case for each query gene and then to combine the resulting clusters in an adequate way.

- **Imitating the Selector's Strategy.** IBEA implements a dominance-based selection scheme where the local search pursues a different strategy in the gene removal and addition procedures. The question arises whether it was beneficial for the local search to imitate the selector more closely. Unfortunately one faces the problem that the local search does not have access to population-wide information that would be needed. Perhaps one could apply a dominance-based scheme that compares the actual and a new solution generated by the local search.

Acknowledgment

At this point I want to grab the chance to thank my supervisor Stefan Bleuler for the many clarifying and helpful comments and questions that he enriched by profound arguments and a great interest from his side in the progress of the work. I highly appreciate that he always took time whenever I went by to ask for assistance; this is not at all to be taken for granted.
Moreover I am indebted to my supervising Professor Eckart Zitzler who offered me the great opportunity for this exciting and very instructive diploma thesis in his research group. If there was a second one to accomplish, I'd be back to his group at any time.

Appendix A

Selected Run Configurations

	qgs_ls_param										qgs_t2_constr			ebica_param						IBEA	PISA_cfg	
run title	input_file_base	query_gene_id	size_obj_mode	max_no_genes	max_no_genes_k	constr_type	distance_measure	do_rank_scaling	max_add_delete_genes	ls_add_genes_q	min_no_genes	min_no_chips 1	min_no_chips 2	seed	p_mut	mutation_type	do_env_selection	maxgen	update	tournament	alpha = mu = lambda	dim
out.13	cold_full_						rank_msr	1	10	.50		6	6		.10	equal_no_bits	0		0	2	100	
out.13a	hetero_full_										32		11									
out.14				200	100				25		10							150				
out.14a		23	additional_constr																			
out.15		3739		60					0													
out.16				100	50				15									100				
out.16a				500	250				0													
out.16b			disable_size_obj						3													2
out.17		23							25		333											
out.18			additional_constr	200	100	linear			0		10							150				3
out.18a			disable_size_obj								24											2
out.18b		23	additional_constr	200	100	exponential					10										400	3
out.19														23				100				
out.19a						linear																
out.19b			disable_size_obj								24											2
out.20			additional_constr			exponential					10			88								3
out.20a						linear																
out.20b			disable_size_obj								24											2
eval.1.1.0-4		23	disable_size_obj	200	100	linear	rank_msr				15			178+k13				150				
eval.1.2.0-4											37			178+k13								
eval.1.3.0-4											58			178+k13								
eval.1.4.0-4											66			178+k13								
eval.2.1.0-4		1313									15			178+k13								
eval.2.2.0-4											37			178+k13								
eval.2.3.0-4											58			178+k13								
eval.2.4.0-4											66			178+k13								
eval.3.0-4		23	additional_constr	200	100	linear	rank_msr				10			178+k13								3
eval.4.0-4		1313																				
eval.5		23	disable_size_obj	200	100	linear	rank_msr							113								2
eval.5a							pearson_corr...															
eval.5b							euclidian															
eval.7			additional_constr				rank_msr							191							1000	3
eval.8														230							400	
eval.8a																						
eval.9			disable_size_obj											113								2
eval.9a																independent_bit						

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 49–57, New York, NY, USA, 2002. ACM Press.

[2] Stefan Bleuler, Marco Laumanns, Lothar Thiele, and Eckart Zitzler. PISA — a platform and programming language independent interface for search algorithms. TIK-Report 154, ETH Zürich, Computer Engineering and Networks Laboratory (TIK), Gloriastrasse 35, ETH-Zentrum, CH-8092 Zürich, Switzerland, October 2002.

[3] Stefan Bleuler, Amela Prelić, and Eckart Zitzler. An EA framework for biclustering of gene expression data. In *Congress on Evolutionary Computation (CEC-2004)*, pages 166–173, Piscataway, NJ, 2004. IEEE.

[4] Stefan Bleuler, Philip Zimmermann, Markus Friberg, Anja Wille, Simon Barkow, Dimo Brockhoff, Daniel Schöner, Lars Hennig, Peter Bühlmann, Wilhelm Gruissem, Lothar Thiele, and Eckart Zitzler. Cluster analysis of multiple time course data sets. Article, ETH Zürich, Computer Engineering and Networks Laboratory (TIK), Gloriastrasse 35, ETH-Zentrum, CH-8092 Zürich, Switzerland, 2006.

[5] Stefan Bleuler and Eckart Zitzler. Order preserving clustering over multiple time course experiments. In *EvoWorkshops 2005*, number 3449 in LNCS, pages 33–43. Springer, 2005.

[6] Yizong Cheng and George M. Church. Biclustering of expression data. In *ISMB*, pages 93–103, 2000.

[7] Seokkyung Chung, Jongeun Jun, and Dennis McLeod. Mining gene expression datasets using density-based clustering. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 150–151, New York, NY, USA, 2004. ACM Press.

[8] Richard Dawkins. *The Selfish Gene.* Oxford University Press, New York, 1976.

[9] Emanuel Falkenauer. *Genetic Algorithms and Grouping Problems.* John Wiley & Sons, Inc., New York, NY, USA, 1998.

[10] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Abstraction and reuse of object-oriented design. *Lecture Notes in Computer Science*, 707:406–431, 1993.

[11] David J. Hand and Nicholas A. Heard. Finding groups in gene expression data. *Journal of Biomedicine and Biotechnology*, pages 215–255, 2005.

[12] Julia Handl and Joshua Knowles. Evolutionary multiobjective clustering. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, 2004.

[13] Julia Handl and Joshua D. Knowles. Exploiting the trade-off - the benefits of multiple objectives in data clustering. In *EMO*, pages 547–560, 2005.

[14] J. A. Hartigan. Direct clustering of a data matrix. In *Journal of the American Statistical Association*, volume 67, pages 123–129, March 1972.

[15] Isaac S. Kohane, Atul J. Butte, and Alvin Kho. *Microarrays for an Integrative Genomics.* MIT Press, Cambridge, MA, USA, 2002.

[16] Peter Merz. Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies, 2000.

[17] Zbigniew Michalewicz and David B. Fogel. *How to solve it: Modern Heuristics.* Springer, 2., rev. and ext. ed. edition, 2004.

[18] P. Moscato and M. G. Norman. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 177–186, Amsterdam, 1992. IOS Press.

[19] Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P 826, Pasadena, CA, 1989.

[20] A. Owen, J. Stuart, K. Mach, A. Villeneuve, and S. Kim. A gene recommender algorithm to identify coexpressed genes in *C. elegans.* In *Genome Res*, pages 1828–1837, 2003.

[21] John Quackenbush. Computational analysis of microarray data. *Nature Reviews Genetics*, pages 418–427, 2001.

[22] Lukas Ruf. *LaTeX Essentials – HowTo Create Your LaTeX-based documents.* Computer Engineering Laboratory TIK, ETH Zurich, 2002.

[23] N. Speer, C. Spieth, and A. Zell. A memetic co-clustering algorithm for gene expression profiles and biological annotation. In *Proceedings of the IEEE 2004 Congress on Evolutionary Computation, CEC 2004*, volume 2, pages 1631–1638. IEEE Press, 2004.

[24] Eckart Zitzler and Anne Auger. *Bioinspired Computation and Optimization.* Lecture script at ETH (Swiss Federal Institute of Technology) Zurich, 2005.

[25] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In Xin Yao et al., editors, *Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, Berlin, Germany, 2004. Springer-Verlag.